Clustering incrémental de signaux audio

Maxime Sirbu

Rapport de stage :

Partitionnement incrémental de signaux audio

Maxime SIRBU

encadré par

Arshia CONT – MuTant, Ircam
Mathieu LAGRANGE – Irccyn

Mars – Juillet 2015

Master ATIAM
UNIVERSITÉ PIERRE ET MARIE CURIE – PARIS 6

Résumé

Ce rapport de stage vise à l'étude de méthodes de partitionnement incrémentales, principalement appliquées à des signaux audio. Nous détaillons tous d'abord les algorithmes de partitionnement classiques de la littérature, ainsi que les bases théoriques permettant d'y aboutir. Puis nous présentons des méthodes étendant ces algorithmes de partitionnement au calcul en ligne. Nous utilisons pour cela les HMM, qui sont une modélisation classique de données et états latents en traitement du signal. Nous utilisons aussi les HSMM, qui sont une extension des HMM, permettant une représentation semi-markovienne des états cachés. Ces algorithmes sont présentés dans le cadre de la segmentation audio – visant à séparer un fichier en fragments homogènes – et de la classification – visant à identifier les différents fragments – pouvant être appliquées à la détection d'événements sonores. On proposera aussi un protocole d'évaluation pour ces méthodes, afin de comparer leurs performances par rapport à une autre méthode de l'état de l'art en leur faisant effectuer les mêmes tâches.

Abstract

This report aims to study different methods of online clustering, mainly applied to audio signals. We will first detail the state-of-the art algorithms for clustering, as well as the theory behind them. Then we will extend this methods to incremental clustering, and present different online algorithms. These algorithms are based on the hidden markov models, which are classic art representations of data and hidden states in signal processing, and hidden semi-markov models, which extend them to a semi-markov representation of the states. We present this within the context of audio segmentation – the task of segmenting audio sources in homogenous chunks – and classification – the task of identifying these chunks – applied to audio event detection. We will also set an experimental protocol, with a view to evaluate them and compare the result to a state-of-the art algorithm for the same task.

Mots-clés : partitionnement, segmentation audio, apprentissage statistique, apprentissage incrémental, modèle semi-markovien caché, détection d'événements audio

Notations

Abréviations

TFCT	–	Transformée de Fourier à Court Terme
MFCC	–	*Mel-Frequency Cepstrum Coefficients*
KL	–	Kullblack-Leibler
IS	–	Itakura-Saito
EM	–	Espérance-Maximisation
EM	–	Espérance-Maximisation
HMM	–	*Hidden Markov Models*
HSMM	–	*Hidden Semi Markov Models*
EBR	–	*Event to Background power Ratio*
v.a.	–	variable aléatoire
i.i.d.	–	indépendantes et identiquement distribuées

Notations mathématiques

$\hat{x}$	–	Transformée de fourier de x
$\mathbf{v}, \mathbf{M}$	–	Un vecteur, une matrice
$\mathbf{M}^{\mathrm{T}}$	–	Transposé de $\mathbf{M}$
$\langle \cdot, \cdot \rangle$	–	Produit scalaire
$\mathbb{P}(\cdot)$	–	Probabilité
$\mathbb{E}[\cdot]$	–	Espérance
$\mathrm{p}(\cdot)$	–	Densité de probabilité
$\mathcal{A}$	–	Un ensemble
$\lvert \cdot \rvert$	–	Cardinal
$\bar{\mathcal{A}}$	–	Fermeture de l'ensemble $\mathcal{A}$

Table des matières

Introduction

Le partitionnement, ou *clustering* en anglais, est une tâche classique en apprentissage statistique. Ce terme regroupe des méthodes permettant de séparer un ensemble de données en différents groupes homogènes et dont les éléments présentent une certaine similarité – tout cela de manière non supervisée. En traitement du signal appliqué au son, ces méthodes peuvent servir à la classification de signaux audio (reconnaissance de phonème, classification de genres musicaux,...).

De plus, ces méthodes peuvent aussi servir à la segmentation de signaux audio. Cette tâche consiste en la séparation d'un signal d'entrée en blocs contigus, de sorte que deux blocs adjacents présentent des contenus audio que l'on jugerait différents à l'écoute. Chaque bloc correspondra alors en général au son émis par une certaine source (*e.g.* note jouée par un instrument à une certaine hauteur, bruit percussif,...), cela peut servir, notamment pour les outils de transcription automatique de note, ou à la détection d'événements sonores particuliers.

Certaines méthodes de segmentation audio n'utilisent cependant pas les méthodes de partitionnement, mais se basent sur la détection de changements abrupts dans le contenu audio, ou *change detection* (Dessein et al. 2013 ; Tzanetakis et al. 1999). Si ces méthodes segmentent bien le signal en différents segments, elles ne permettent cependant pas de détecter les similarités entre blocs non contigus. Il faut alors les calculer après la segmentation, en utilisant des outils tels que des matrices de similarité. On peut cependant citer Cont et al. (2011) et Lostanlen (2013), qui proposent une méthode de segmentation permettant de calculer une mesure de similarité entre segments, basée sur la géométrie de l'information. Ces méthodes sont supervisées dans la plupart des cas : le bon fonctionnement des algorithmes implique donc une connaissance *a priori* du signal audio. Cela se fait en utilisant une base de données d'apprentissage, ce qui implique que les performances des algorithmes dépendent fortement de l'apprentissage, et cela en fait des méthodes peu robustes à la détection d'événements nouveaux. Cependant elles peuvent être en général utilisées en ligne : l'algorithme considère les données au fur et à mesure que celles-ci sont disponibles, et non pas l'ensemble des données en tant que lot (ou *batch*). Cela présente un avantage certain pour des applications en temps réel.

Nous proposons ici d'utiliser au contraire des méthodes de partitionnement plus classiques pour la segmentation de signaux audio, permettant de segmenter et classifier les blocs dans un même processus. Ces méthodes présentent l'avantage d'être non supervisées. Nous utiliserons pour cela les modèles cachés semi-markoviens (hsmm), qui sont une extension des modèles cachés markoviens classiques (hmm) (Guédon 2003 ; Yu 2010). Cependant ces modèles opèrent classiquement de manière hors-ligne, nous utilisons alors des extensions des algorithmes pour le calcul en ligne, introduit par (Bietti 2014). Ceux-ci proposent un calcul des valeurs d'intérêt de manière récursive, amenant la possibilité d'une mise à jour de ces valeurs à partir des données à l'instant du calcul. On parle alors aussi d'algorithme d'apprentissage *incrémental*.

Le partitionnement audio incrémental présente un intérêt majeur, en effet il ouvre la voie à des applications temps réel. On peut par exemple citer le logiciel de suivi de partition *Antescofo*. Celui-ci se base actuellement sur des modèles (*template*) pour chaque note et accord observés. Les méthodes proposées pourraient permettre un apprentissage plus robuste, et étendre la détection à des événements sonores non stationnaires.

Dans le chapitre I, nous présenterons les méthodes de classification et partitionnement classiques, appliquées dans le cas de l'audio. Nous introduirons aussi le concept de divergences de Bregman, per-

mettant d'utiliser de nombreuses mesures de similarités différentes pour les algorithmes (Banerjee, Guo et al. 2004).

Dans le chapitre II, nous présenterons des algrothimes étendant les méthodes hors-ligne à un calcul en ligne du partitionnement. Plus précisément nous introduirons les travaux de Bietti et al. (2015), proposant des algorithmes en ligne pour hsmm, basés sur Cappé et Moulines (2009) et Neal et al. (1998).

Les algorithmes présentés feront l'objet d'une évaluation dans le chapitre III. Nous introduirons l'implémentation de ces méthodes, ainsi que quelques résultats succincts sur des exemples simples. Nous évaluerons aussi les algorithmes en utilisant des métriques appropriées, en comparant les résultats obtenus à des valeurs de références de l'état de l'art.

Ce travail de stage a été l'occasion de mettre en application les résultats théoriques obtenus par Bietti, stagiaire dans l'équipe MuTant de l'Ircam en 2014, et ayant proposé les algorithmes en ligne pour hsmm présentés dans ce rapport. Tout d'abord en travaillant sur l'implémentation des algorithmes, puis en les évaluant sur des jeux de données de l'état de l'art. Plus précisément, les principales contributions ont été :

- Remaniement formel des méthodes, notamment pour l'apprentissage des durées de séquences du modèle hsmm (voir section I-6.4).
- Travail sur l'implémentation des méthodes de partitionnement, hors-ligne et en ligne à l'aide du language `python` et du cadriciel de calcul scientifique `numpy/scipy`. L'ensemble des méthodes présentées ont été developpées en reprenant le code de Bietti. Les algorithmes déjà implémentés ont été revus et corrigés, et leur performances améliorées.
- Mise en place d'un protocole et d'un cadre d'évaluation des algorithmes, en utilisant les mêmes métriques qu'une référence concurrente afin de comparer les performances des méthodes proposées face à des résultats préexistants. Analyse des différents résultats obtenus.

Chapitre I

Modélisation et partitionnement hors-ligne

Le partitionnement de signaux est un problème très fréquent en traitement du signal et en apprentissage statistique. De nombreux modèles et algorithmes ont été mis en œuvre pour le résoudre dans de nombreux contextes différents.

Nous nous appuyons ici tout d'abord sur le modèle proposé par Banerjee, Merugu et al. qui utilise les divergences de Bregman(Banerjee, Merugu et al. 2005). Celui-ci énonce deux algorithmes pour résoudre ce problème, en utilisant les divergences de Bregman : l'algorithme des k-moyennes, et l'algorithme Espérance-Maximisation appliqué aux modèles de mélange. Nous montrerons ensuite comment utiliser les modèles de markov cachés (hmm), offrant une modélisation plus performante de signaux, et nous introduirons enfin les modèles semi-markoviens cachés (Guédon 2003 ; Yu 2010)

1. Divergences de Bregman

1. Mesures de similarité

Les algorithmes de *clustering* sont utilisés pour effectuer des tâches de classification ou de segmentation. Dans tous les cas, ils ont pour but la séparation d'atomes en différents groupes homogènes. Pour ce faire on doit utiliser une mesure de similarité afin de comparer les atomes deux à deux. Si l'on cherche à partitionner des éléments appartenant à un espace $\mathbb{R}^d, d \in \mathbb{N}^*$, une mesure de similarité triviale est la distance euclidienne : $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2$

Cependant, le fait d'utiliser la distance euclidienne comme mesure de similarité implique un espace à géométrie euclidienne, ce qui n'est pas le cas pour les représentations temps-fréquence de signaux audio. En traitement du signal appliqué à l'audio, on préfèrera en général utiliser des mesures alternatives telles que les divergences de Kullblack-Leibler (kl) et de Itakura-Saito (is) (Cont et al. 2011 ; Kemp et al. 2000) :

Définition I.1 Divergence de Kullblack-Leibler

Soit deux vecteurs $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ tels que $\mathbf{x}$ et $\mathbf{y}$ représentent des distributions de probabilité discrètes, *i.e.* $\sum_{i=1}^d x_i = \sum_{i=1}^d y_i = 1$, on définit alors la kl-divergence entre $\mathbf{x}$ et $\mathbf{y}$ par :

$$d_{\text{KL}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} x_i \ln(\frac{x_i}{y_i})$$

Définition I.2 Divergence de Itakura-Saito

Soit deux vecteurs $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, on définit alors la is-divergence entre $\mathbf{x}$ et $\mathbf{y}$ par :

$$d_{\text{IS}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} \frac{x_i}{y_i} - \ln(\frac{x_i}{y_i}) - 1$$

On remarquera que ces divergences ne sont en revanche pas des distances au sens mathématique du terme. Elles possèdent bien la propriété de séparation $d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$, en revanche elles ne respectent ni la symétrie $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$, ni l'inégalité triangulaire.

2. Divergences de Bregman

La plupart des algorithmes de *clustering* peuvent être adaptés pour utiliser l'une de ces métriques comme mesure de similarité. Banerjee, Merugu et al. (2005) montre cependant que ces divergences peuvent être généralisées par le concept de divergence de Bregman (Bregman 1967) :

Définition I.3 Divergence de Bregman

Soit $\phi : \Omega \mapsto \mathbb{R}$ une fonction strictement convexe définie sur un ensemble convexe $\Omega \subset \mathbb{R}^d$ et différentiable, on définit la divergence de Bregman $d_\phi : \Omega^2 \mapsto [0, +\infty[$ par :

$$d_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - \langle \mathbf{x} - \mathbf{y}, \nabla\phi(\mathbf{y}) \rangle \tag{I.1}$$

où $\nabla\phi(\mathbf{y})$ est le gradient de ϕ en $\mathbf{y}$.

Exemple I.1 Distance Euclidienne

La distance euclidienne est l'exemple le plus simple de divergence de Bregman. On pose $\phi(\mathbf{x}) = \langle \mathbf{x}, \mathbf{x} \rangle$, définie et différentiable sur $\mathbb{R}^d$.

$$\begin{aligned} d_\phi(\mathbf{x}, \mathbf{y}) &= \phi(\mathbf{x}) - \phi(\mathbf{y}) - \langle \mathbf{x} - \mathbf{y}, \nabla\phi(\mathbf{y}) \rangle \\ &= \langle \mathbf{x}, \mathbf{x} \rangle - \langle \mathbf{y}, \mathbf{y} \rangle - \langle \mathbf{x} - \mathbf{y}, 2\mathbf{y} \rangle \\ &= \langle \mathbf{x}, \mathbf{x} \rangle + \langle \mathbf{y}, \mathbf{y} \rangle - 2 \langle \mathbf{x}, \mathbf{y} \rangle \\ &= \langle \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} \rangle = \|\mathbf{x} - \mathbf{y}\|^2 \end{aligned}$$

Exemple I.2 KL-divergence

On définit pour $\mathbf{x} \in \{\mathbf{x} | \sum_i x_i = 1\}$, $\phi(\mathbf{x}) = \sum_i x_i \ln x_i$.

$$\begin{aligned} d_\phi(\mathbf{x}, \mathbf{y}) &= \phi(\mathbf{x}) - \phi(\mathbf{y}) - \langle \mathbf{x} - \mathbf{y}, \nabla\phi(\mathbf{y}) \rangle \\ &= \sum_i x_i \ln x_i - \sum_i y_i \ln y_i - \sum_i (x_i - y_i)(\ln y_i - 1) \\ &= \sum_i x_i (\ln x_i - \ln y_i) - \sum_i x_i + \sum_i y_i \\ &= \sum_i x_i \ln \frac{x_i}{y_i} \\ &= d_{\mathrm{KL}}(\mathbf{x}, \mathbf{y}) \end{aligned}$$

Exemple I.3 IS-divergence

Soit $\phi(\mathbf{x}) = -\sum_i \ln x_i$ définie sur $\mathbb{R}^{+*d}$.

$$\begin{aligned} d_\phi(\mathbf{x}, \mathbf{y}) &= \phi(\mathbf{x}) - \phi(\mathbf{y}) - \langle \mathbf{x} - \mathbf{y}, \nabla\phi(\mathbf{y}) \rangle \\ &= -\sum_i \ln x_i + \sum_i \ln y_i + \sum_i (x_i - y_i) \frac{1}{y_i} \\ &= \sum_i \frac{x_i}{y_i} - \ln \frac{x_i}{y_i} - 1 \\ &= d_{\mathrm{IS}}(\mathbf{x}, \mathbf{y}) \end{aligned}$$

2. Partitionnement « Dur »

1. Centroïde

Si on pose un ensemble de points $\{\mathbf{x}_i\}_{i=1}^n \subset \Omega \subseteq \mathbb{R}^d$, on peut poser le problème du meilleur représentant, ou plus proche voisin de l'ensemble de ces points. Intuitivement on considère que le centroïde de ces points, défini par $\boldsymbol{\mu} = \frac{1}{n}\sum_i \mathbf{x}_i \in \Omega$ (car Ω est convexe), est le meilleur représentant.

On peut montrer que le centroïde correspond bien au plus proche voisin, non seulement au sens de la distance euclidienne, mais aussi au sens de toute divergence de Bregman définie par (I.1) (Banerjee, Merugu et al. 2005 ; Nielsen et al. 2009).

Proposition I.1 *Soit un v.a. X à valeurs dans un ensemble $\{\mathbf{x}_i\}_{i=1}^n \subset \Omega$ et suivant une mesure discrète ν et soit d_ϕ une divergence de Bregman définie par (I.1). On pose $\boldsymbol{\mu} \mathrel{\widehat{=}} \mathbb{E}_\nu[X] = \sum_i \nu_i \mathbf{x}_i$. On a alors :*

$$\boldsymbol{\mu} = \arg\min_{\mathbf{c}\in\Omega} \mathbb{E}_\nu[d_\phi(X, \mathbf{c})] \tag{I.2}$$

Preuve :

$$\begin{aligned}
\mathbb{E}_\nu[d(X,\mathbf{c})] &= \sum_i \nu_i d_\phi(\mathbf{x}_i, \mathbf{c}) \\
&= \sum_i \nu_i (\phi(\mathbf{x}_i) - \phi(\mathbf{c}) - \langle \mathbf{x}_i - \mathbf{c}, \nabla\phi(\mathbf{c})\rangle \\
&= \mathbb{E}_\nu[\phi(X)] - \phi(\mathbf{c}) - \langle \boldsymbol{\mu} - \mathbf{c}, \nabla\phi(\mathbf{c})\rangle \\
&= \mathbb{E}_\nu[\phi(X)] + d_\phi(\boldsymbol{\mu}, \mathbf{c}) - \phi(\boldsymbol{\mu})
\end{aligned}$$

Minimal pour $d_\phi(\boldsymbol{\mu}, \mathbf{c}) = 0$, *i.e.* $\mathbf{c} = \boldsymbol{\mu}$. □

On a généralisé ici à tout centroïde pondéré, pour les problèmes de partitionnement, on se restreindra en général au cas $\nu_i = \frac{1}{n}, \forall i \in [\![1; n]\!]$.

Il faut aussi bien prendre en compte le fait que $\boldsymbol{\mu}$ ne minimise en général que $\mathbb{E}[d_\phi(X, \mathbf{c})]$ et pas $\mathbb{E}[d_\phi(\mathbf{c}, X)]$ puisque les divergences de Bregman ne sont pas forcément symétriques. On appelle donc $\boldsymbol{\mu}$ le centroïde de type droit. Nielsen et al. (2009) définissent aussi un centroïde de type gauche, ainsi qu'un centroïde symétrique.

2. Information de Bregman

La valeur minimisée par le centroïde est dénommé information de Bregman par Banerjee, Merugu et al. (2005).

Définition I.4 Information de Bregman

Soit d_ϕ une divergence de Bregman et X une v.a. à valeurs dans $\{\mathbf{x}_i\}_{i=1}^n \subset \Omega$ et suivant une mesure discrète ν, l'information de Bregman associée à ϕ est :

$$\mathrm{I}_\phi^\nu(X) = \min_{\mathbf{c}\in\Omega} \mathbb{E}_\nu[d_\phi(X, \mathbf{c})] = \mathbb{E}_\nu[d_\phi(X, \boldsymbol{\mu})] \tag{I.3}$$

Avec $\boldsymbol{\mu}$ défini par (I.2).

L'information de Bregman peut prendre différentes significations selon le type de divergence utilisée :

Distance Euclidienne Pour $d_\phi(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2$, $\mathrm{I}_\phi^\nu(X) = \mathbb{E}_\nu\left[\|\mathbf{x}_i - \boldsymbol{\mu}\|^2\right]$, ce qui correspond à la variance de X.

Divergence de Kullblack-Leibler Si on pose $\nu_i = \mathrm{p}(U = u_i)$ la probabilité de réalisation d'une v.a. U discrète dans $\{u_i\}_{i=1}^n$, et $\mathbf{x}_i = \mathrm{p}(V|u_i)$ la probabilité conditionnelle d'une autre v.a. discrète V à valeurs dans $\{v_i\}_{i=1}^m$, alors $\boldsymbol{\mu} = \sum_i \mathrm{p}(u_i)\mathrm{p}(V|u_i) = \mathrm{p}(V)$ et

$$\begin{aligned}
\mathrm{I}_\phi^\nu(X) &= \sum_{i=1}^{n} \mathrm{p}(u_i) * d_{\mathrm{KL}}(\mathrm{p}(V|u_i), \mathrm{p}(V)) \\
&= \sum_{i=1}^{n} \mathrm{p}(u_i) \sum_{j=1}^{m} \mathrm{p}(v_j|u_i) \ln \frac{\mathrm{p}(v_j|u_i)}{\mathrm{p}(v_j)} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{m} \mathrm{p}(u_i, v_j) \ln \frac{\mathrm{p}(u_i, v_j)}{\mathrm{p}(u_i)\mathrm{p}(v_j)} \\
&= I(U, V)
\end{aligned}$$

où $I(U, V)$ est l'information mutuelle entre les variables U et V (Dhillon et al. 2003 ; Kraskov et al. 2003).

On peut aussi remarquer que l'information de Bregman correspond à la différence entre les termes de l'inégalité de Jensen : $\mathbb{E}[\phi(X)] \geq \phi(\mathbb{E}[X])$ pour toute fonction ϕ convexe et toute variable aléatoire X (Banerjee, Guo et al. 2004).

$$\begin{aligned}
\mathbb{E}[\phi(X)] - \phi(\mathbb{E}[X]) &= \mathbb{E}[\phi(X)] - \phi(\mathbb{E}[X]) - \overbrace{\mathbb{E}[\langle X - \mathbb{E}[X], \nabla\phi(\mathbb{E}[X])\rangle]}^{=0} \\
&= \mathbb{E}[\phi(X) - \phi(\mathbb{E}[X]) - \langle X - \mathbb{E}[X], \nabla\phi(\mathbb{E}(X))\rangle] \\
&= \mathbb{E}[d_\phi(X, \mathbb{E}[X])] = \mathrm{I}_\phi(X) \geq 0
\end{aligned}$$

3. Partitionnement

Partitionner un corpus défini par une variable aléatoire X sur $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ en K classes, revient à définir K v.a. $\{X_h\}_{h=1}^K$ définies sur les sous-ensembles $\{\mathcal{X}_h\}_{h=1}^K$, tels que $\bigcup_h \mathcal{X}_h = \mathcal{X}$ et $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset \; \forall i \neq j$. Chaque v.a. X_h suit alors une mesure $\epsilon_h = \frac{\nu}{\pi_h}$ avec $\pi_h = \sum_{i \in \{i|\mathbf{x}_i \in \mathcal{X}_h\}} \nu_i$, et on peut définir les K meilleurs représentants $\mathcal{M} = \{\boldsymbol{\mu}_h\}_{h=1}^K$ de chaque partition comme en (I.2).

Pour que le partitionnement soit bon, il faut que les centroïdes $\{\boldsymbol{\mu}_h\}_{h=1}^K$ soient les plus proches possible des éléments de chaque $\mathcal{X}_h$ selon la divergence de Bregman d_ϕ. On peut définir une autre v.a. M à valeurs dans $\mathcal{M}$ suivant la mesure implicite π définie précédemment, on va alors chercher à minimiser $\mathbb{E}_{\nu,\pi}[d_\phi(X, M)]$. Comme M est une fonction déterministe de X, on a $\mathbb{E}_{\nu,\pi}[d_\phi(X, M)] = \mathbb{E}_\nu[d_\phi(X, M)]$ et

$$\mathbb{E}_\nu[d_\phi(X, M)] = \sum_{h=1}^{K} \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h) = \sum_{h=1}^{K} \pi_h \sum_{\mathbf{x}_i \in \mathcal{X}_h} \frac{\nu_i}{\pi_h} d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h) = \mathbb{E}_\pi\left[\mathrm{I}_\phi^{\epsilon_H}(X_H)\right] \tag{I.4}$$

Avec H v.a. à valeurs dans $[\![1; K]\!]$ suivant la mesure π. En d'autres termes, trouver le meilleur partitionnement revient à minimiser l'espérance de l'information de Bregman de chaque partition.

Si l'on reprend la distance euclidienne comme exemple de divergence de Bregman, (I.4) correspond à l'espérance de la variance des partitions, minimiser cette espérance correspond bien à une approche intuitive euclidienne du partitionnement, où l'on cherche à avoir des partitions de faible variance.

Une autre façon de concevoir le partitionnement, plus basée sur la théorie de l'information, consiste à minimiser la perte d'information engendrée – en général la perte d'information mutuelle entre classes et observations due à la quantification (Dhillon et al. 2003). Ici on peut considérer

la perte d'information de Bregman $\mathrm{I}_\phi^\nu(X) - \mathrm{I}_\phi^\pi(M)$ comme mesure de qualité du partitionnement. Si $K = n$ le nombre d'observations, choisir $\mathcal{X}_h = \{\mathbf{x}_h\}$ donne $\mathrm{I}_\phi^\nu(X) = \mathrm{I}_\phi^\pi(M)$, donc une perte nulle. Si $K = 1$, alors $\mathcal{M} = \{\mathbb{E}_\nu[X]\}$, donc $\mathrm{I}_\phi^\pi(M) = 0$ et on a une perte de $\mathrm{I}_\phi^\nu(X)$.

On peut montrer que cette nouvelle métrique de qualité du partitionnement est exactement égale à la précédente.

Proposition I.2 *Soit X une v.a. discrète sur $\mathcal{X} = \{x_i\}_{i=1}^n$ suivant la mesure ν, et un partitionnement $\{\mathcal{X}_h\}_{h=1}^K$ de $\mathcal{X}$ impliquant la mesure $\pi_h = \sum_{i\in\{i|x_i\in\mathcal{X}_h\}} \nu_i$. Pour tout $h \in [\![1; K]\!]$, soit X_h la v.a. à valeurs dans $\mathcal{X}_h$ suivant la mesure $\epsilon_h = \frac{\nu}{\pi_h}$, et $\boldsymbol{\mu}_h = \mathbb{E}_{\epsilon_h}[X_h]$. On définit enfin M et H, v.a. à valeur dans $\mathcal{M} = \{\boldsymbol{\mu}_h\}_{h=1}^K$ et $[\![1, K]\!]$ respectivement, suivant la mesure implicite π. On peut définir la* fonction de perte *du partitionnement,*

$$\mathcal{L}_\phi(X, X_H) = \mathrm{I}_\phi^\nu(X) - \mathrm{I}_\phi^\pi(M) = \mathbb{E}_\pi\left[\mathrm{I}_\phi^{\epsilon_H}(X_H)\right] \tag{I.5}$$

Preuve : Par définition

$$\begin{aligned}
\mathrm{I}_\phi^\nu(X) &= \sum_{i=1}^n \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}) = \sum_{h=1}^K \sum_{\mathbf{x}_i\in\mathcal{X}_h} \nu_i \left[\phi(\mathbf{x}_i) - \phi(\boldsymbol{\mu}) - \langle \mathbf{x}_i - \boldsymbol{\mu}, \nabla\phi(\boldsymbol{\mu})\rangle\right] \\
&= \sum_{h=1}^K \sum_{\mathbf{x}_i\in\mathcal{X}_h} \nu_i \left[\phi(\mathbf{x}_i) - \phi(\boldsymbol{\mu}_h) - \langle \mathbf{x}_i - \boldsymbol{\mu}_h, \nabla\phi(\boldsymbol{\mu}_h)\rangle \right. \\
&\qquad \left. + \phi(\boldsymbol{\mu}_h) - \phi(\boldsymbol{\mu}) - \langle \boldsymbol{\mu}_h - \boldsymbol{\mu}, \nabla\phi(\boldsymbol{\mu})\rangle + \langle \mathbf{x}_i - \boldsymbol{\mu}_h, \nabla\phi(\boldsymbol{\mu}_h) - \nabla\phi(\boldsymbol{\mu})\rangle\right] \\
&= \sum_{h=1}^K \sum_{\mathbf{x}_i\in\mathcal{X}_h} \nu_i \left[d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h) + d_\phi(\boldsymbol{\mu}_h, \boldsymbol{\mu}) + \langle \mathbf{x}_i - \boldsymbol{\mu}_h, \nabla\phi(\boldsymbol{\mu}_h) - \nabla\phi(\boldsymbol{\mu})\rangle\right] \\
&= \sum_{h=1}^K \pi_h \left[\sum_{\mathbf{x}_i\in\mathcal{X}_h} \frac{\nu_i}{\pi_h} d_\phi(\mathbf{x}_i, \boldsymbol{\mu}) + d_\phi(\boldsymbol{\mu}_h, \boldsymbol{\mu}) + \left\langle \underbrace{\sum_{\mathbf{x}_i\in\mathcal{X}_h} \frac{\nu_i}{\pi_h} x_i}_{=\boldsymbol{\mu}_h} - \boldsymbol{\mu}_h, \nabla\phi(\boldsymbol{\mu}_h) - \boldsymbol{\mu}\phi(\boldsymbol{\mu})\right\rangle\right] \\
&= \sum_{h=1}^K \pi_h \mathrm{I}_\phi^{\epsilon_h}(X_h) + \sum_{h=1}^K \pi_h d_\phi(\boldsymbol{\mu}_h, \boldsymbol{\mu}) \\
&= \mathbb{E}_\pi\left[\mathrm{I}_\phi^{\epsilon_H}(X_H)\right] + \mathrm{I}_\phi^\pi(M)
\end{aligned}$$

D'où (I.5). □

On peut remarquer que l'information de Bregman totale équivaut donc à l'information de Bregman entre-clusters ($\mathrm{I}_\phi^\pi(M)$), plus l'information de Bregman intra-clusters ($\mathbb{E}_\pi\left[\mathrm{I}_\phi^{\epsilon_H}(X_H)\right]$).

4. Algorithme des K-Moyennes

On peut déduire de la proposition ci-dessus une généralisation de l'algorithme de K-Moyennes (MacQueen 1967) aux divergences de Bregman. Cet algorithme est classiquement basé sur une métrique euclidienne, et permet de diminuer la fonction de coût en deux temps :

— Assignation de chaque élément à la partition dont le représentant est le plus proche.

— Ré-estimation des représentants de chaque partition à partir des éléments qu'elle contient.

Proposition I.3 *L'algorithme des K-Moyennes fait décroître la fonction de perte $\mathcal{L}_\phi$ (I.5) à chaque itération de manière monotone, et converge en un nombre fini d'étapes.*

Algorithme 1 K-Moyennes

Procédure K-Moyennes($\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \Omega$, ν,d_ϕ,K)
  Initialise $\{\boldsymbol{\mu}_h\}_{h=1}^K$, généralement $\forall h \in [\![1;K]\!], \boldsymbol{\mu}_h \in \mathcal{X}$ choisis aléatoirement
  Répète
    $\mathcal{X}_h^{(\tau+1)} = \emptyset, \forall h \in [\![1;K]\!]$
    Pour $i \leftarrow 1, \dots, n$ **faire**
      $\mathcal{X}_{h_i}^{(\tau+1)} \leftarrow \mathcal{X}_{h_i}^{(\tau+1)} \cup \{\mathbf{x}_i\}$, avec $h_i \leftarrow \arg\min_h \{d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h^{(\tau)})\}\ \forall \mathbf{x}_i \in \mathcal{X}$
    Fin Pour
    Pour $h \leftarrow 1, \dots, K$ **faire**
      $\pi_h^{(\tau+1)} \leftarrow \sum_{\mathbf{x}_i \in \mathcal{X}_h^{(\tau+1)}} \nu_i$
      $\boldsymbol{\mu}_h^{(\tau+1)} \leftarrow \frac{1}{\pi_h^{(\tau+1)}} \sum_{\mathbf{x}_i \in \mathcal{X}_h^{(\tau+1)}} \nu_i x_i$
    Fin Pour
  Jusqu'à convergence
  Retourne $\{\mathcal{X}_h\}_{h=1}^K$
Fin Procédure

Preuve : Soit $\mathcal{X}_h^{(\tau)}$ et $\boldsymbol{\mu}_h^{t)}$, respectivement la partition h son centroïde à l'itération t, alors

$$\begin{aligned}\mathcal{L}_\phi(X, X_H^{(\tau)}) &= \sum_{h=1}^K \sum_{\mathbf{x}_i \in \mathcal{X}_h^{(\tau)}} \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h^{(\tau)}) \geq \sum_{h=1}^K \sum_{\mathbf{x}_i \in \mathcal{X}_h^{(\tau+1)}} \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h^{(\tau)}) \\ &\geq \sum_{h=1}^K \sum_{\mathbf{x}_i \in \mathcal{X}_h^{(\tau)}} \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h^{(\tau+1)}) = \mathcal{L}_\phi(X, X_H^{(\tau+1)})\end{aligned}$$

La première inégalité est due à l'étape d'assignation, puisque par définition si $\mathbf{x}_i \in \mathcal{X}_h^{(\tau)}$ et $\mathbf{x}_i \in \mathcal{X}_{h'}^{(\tau+1)}$, $d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h^{(\tau)}) \geq d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_{h'}^{(\tau)}) = \min_k \{d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_k^{(\tau)})\}$. La deuxième inégalité est due à la proposition I.1. Donc $t \to \mathcal{L}_\phi(X, X_H^{(\tau)})$ est une fonction décroissante, minorée par zéro, donc convergente. Puisqu'il n'existe qu'un nombre fini de partions différentes possible, le nombre d'itérations est lui aussi fini. □

3. Familles exponentielle et divergence de Bregman

Bien souvent, l'algorithme des K-moyenne agit de manière trop rigide. En effet, les tâches d'assignation et d'estimation des centroïdes fonctionnent « en dur », c'est-à-dire sans considérer l'ensemble des assignations possibles, mais en se contentant de choisir la plus performante au sens de la fonction de coût. On utilise souvent des algorithmes basés sur des modèles probabilistes tels que l'algorithme Espérance-Maximisation (EM, *Expectation-Maximization*) pour faire face au manque de souplesse des K-Moyennes (Dempster et al. 1977). Cet algorithme se base sur un modèle de mélange (voir section I-4.2). Banerjee, Merugu et al. (2005) établissent une bijection entre famille exponentielle et divergence de Bregman, permettant de proposer un modèle de mélange probabiliste basé sur cette métrique, et de généraliser l'algorithme EM.

1. Familles exponentielles

Définition I.5 Familles exponentielles

On appelle famille exponentielle toute famille de distribution paramétrique $\mathcal{F}_\psi$ sur un espace mesurable $(\Omega, \mathcal{A})$, telle que

$$\mathcal{F}_\psi = \{\mathrm{p}_{(\psi,\boldsymbol{\eta})} | \boldsymbol{\eta} \in \Theta \subseteq \mathbb{R}^d\} \quad \text{avec} \;\; \mathrm{p}_{(\psi,\boldsymbol{\eta})}(w) = \exp(\langle \mathbf{T}(w), \boldsymbol{\eta}\rangle - \psi(\boldsymbol{\eta}))a(w), \; \forall w \in \Omega$$

où $\mathbf{T} : \Omega \mapsto \mathbb{R}^d$ dénote une statistique exhaustive, $\boldsymbol{\eta}$ est le paramètre naturel de la distribution. La fonction ψ appelée log-partition est définie par $\psi(\boldsymbol{\eta}) = \int_{w\in\Omega} \exp(\langle \mathbf{T}(w), \theta\rangle)a(w)\,\mathrm{d}w$ et permet de vérifier $\int_{w\in\Omega} \mathrm{p}_{(\psi,\boldsymbol{\eta})}\,\mathrm{d}w = 1$. Le tableau I.1 présente quelques familles exponentielles classiques.

Distribution	$\mathrm{p}_{(\psi,\boldsymbol{\eta})}(w)$	$\boldsymbol{\eta}$	$\mathbf{T}(w)$	$\psi(\boldsymbol{\eta})$
$\mathcal{B}(p)$	$p^w(1-p)^{1-w}$	$\ln(\frac{p}{1-p})$	w	$-\ln(1-p)$
$\mathcal{N}(\mu,\sigma)$	$\frac{1}{\sqrt{2\pi\sigma^2}}\mathrm{e}^{-\frac{(w-\mu)^2}{2\sigma^2}}$	$[\frac{\mu}{\sigma^2}, -\frac{\mu^2}{2\sigma^2}]$	$[w, w^2]$	$\frac{\mu^2}{2\sigma^2}$
$\mathcal{P}(\lambda)$	$\frac{\lambda^w \mathrm{e}^{-\lambda}}{w!}$	$\ln(\lambda)$	w	λ
$\mathcal{E}(\lambda)$	$\lambda \mathrm{e}^{-\lambda w}$	$-\lambda$	w	$-\ln(\lambda)$

TABLE I.1 – Familles exponentielles usuelles

La statistique exhaustive $\mathbf{T}(w)$ est dite minimale si $\exists \mathbf{a} \neq \mathbf{0}$ telle que $\langle \mathbf{T}(w), \mathbf{a}\rangle = c \in \mathbb{R}^d$ constant $\forall w \in \Omega$. Afin d'effectuer le rapprochement avec les divergences de Bregman, BANERJEE, GUO et al. (2004) définissent un ensemble plus restreint de familles exponentielles.

Définition I.6 Familles exponentielles régulières

Une famille exponentielle régulière est une famille exponentielle pour laquelle l'espace paramétrique Θ est ouvert, et pour laquelle $\mathbf{x} \in \mathbb{R}^d$ dénote une statistique exhaustive minimale,

$$\mathrm{p}_{(\psi,\boldsymbol{\eta})}(\mathbf{x}) = \exp(\langle \mathbf{x}, \boldsymbol{\eta}\rangle - \psi(\boldsymbol{\eta}))a(\mathbf{x}), \quad \forall \mathrm{p}_{(\psi,\boldsymbol{\eta})} \in \mathcal{F}_\psi \tag{I.6}$$

2. Conjugué de Legendre

La fonction de log-partition $\psi(\boldsymbol{\eta})$ est uniquement déterminée par la famille exponentielle $\mathcal{F}_\psi$ à une constante additive près. On peut montrer que cette fonction est convexe sur l'ensemble convexe Θ, et qu'elle est différentiable. Cela nous permet de caractériser ψ comme une fonction de Legendre, dont les propriétés nous permettront de mettre en relation toute famille exponentielle avec une divergence de Bregman.

Définition I.7 Fonction de Legendre

Soit la fonction ψ définie sur l'ensemble convexe $\Theta \neq \emptyset$, on dit que (ψ, Θ) est une fonction de Legendre si :

1. ψ est convexe et différentiable sur Θ
2. $\forall \boldsymbol{\eta}_l \in \bar{\Theta} - \Theta, \quad \lim_{\boldsymbol{\eta}\to\boldsymbol{\eta}_l} \|\psi(\boldsymbol{\eta})\| = +\infty$, où $\bar{\Theta} - \Theta$ désigne la frontière de Θ.

Définition I.8 Conjugué de Legendre (ROCKAFELLAR 1970)

Soit (ψ, Θ) une fonction de Legendre strictement convexe, on définit son conjugué par :

$$\psi^*(\mathbf{t}) = \sup_{\boldsymbol{\eta}\in\Theta} \langle \mathbf{t}, \boldsymbol{\eta}\rangle - \psi(\boldsymbol{\eta}) \quad \text{définie sur } \Theta^* \tag{I.7}$$

Alors (ψ^*, Θ^*) est aussi une fonction de Legendre, et (ψ, Θ) et (ψ^*, Θ^*) sont appelés conjugués de Legendre.

Puisque ψ est une fonction convexe, on peut obtenir la borne supérieure définie par (I.7) en trouvant la valeur qui annule le gradient de l'expression.

$$\nabla(\langle \mathbf{t}, \boldsymbol{\eta} \rangle - \psi(\boldsymbol{\eta}))_{|\boldsymbol{\eta}_{sup}} = \mathbf{t} - \nabla\psi(\boldsymbol{\eta}_{sup}) = 0 \quad \Rightarrow \quad \mathbf{t} = \nabla\psi(\boldsymbol{\eta}_{sup})$$

Puisque ψ est strictement convexe, $\nabla\psi$ est strictement monotone et donc inversible. On a donc $\boldsymbol{\eta}_{sup} = (\nabla\psi)^{-1}(\mathbf{t})$. On en conclut donc que

$$\psi^*(\mathbf{t}) = \left\langle \mathbf{t}, (\nabla\psi)^{-1}(\mathbf{t}) \right\rangle - \psi((\nabla\psi)^{-}1(\mathbf{t})) \tag{I.8}$$

. Ceci amène à la proposition suivante :

Proposition I.4 *Soit (ψ, Θ) et (ψ^*, Θ^*) deux conjugués de Legendre tels que décrits par la définition I.8. Alors (a) $\psi^{**} = \psi$ et (b) $\nabla\psi^* = (\nabla\psi)^{-1}$.*

Preuve : D'après (I.8) :

(a)
$$\begin{aligned}
\psi^{**}(\boldsymbol{\eta}) &= \left\langle \boldsymbol{\eta}, (\nabla\psi^*)^{-1}(\boldsymbol{\eta}) \right\rangle - \psi^*((\nabla\psi^*)^{-1}(\boldsymbol{\eta})) \\
&= \langle \boldsymbol{\eta}, \nabla\psi(\boldsymbol{\eta}) \rangle - \psi^*(\nabla\psi(\boldsymbol{\eta})) \\
&= \langle \boldsymbol{\eta}, \nabla\psi(\boldsymbol{\eta}) \rangle - \left\langle \nabla\psi(\boldsymbol{\eta}), (\nabla\psi)^{-1}(\nabla\psi(\boldsymbol{\eta})) \right\rangle + \psi((\nabla\psi)^{-1}(\nabla\psi(\boldsymbol{\eta}))) \\
&= \langle \boldsymbol{\eta}, \nabla\psi(\boldsymbol{\eta}) \rangle - \langle \nabla\psi(\boldsymbol{\eta}), \boldsymbol{\eta} \rangle + \psi(\boldsymbol{\eta}) = \psi(\boldsymbol{\eta})
\end{aligned}$$

(b)
$$\begin{aligned}
\nabla\psi^*(\mathbf{t}) &= (\nabla\psi)^{-1}(\mathbf{t}) + \left\langle \mathbf{t}, \nabla(\nabla\psi)^{-1}(\mathbf{t}) \right\rangle - \left\langle \nabla(\nabla\psi)^{-1}(\mathbf{t}), \nabla\psi((\nabla\psi)^{-1}(\mathbf{t})) \right\rangle \\
&= (\nabla\psi)^{-1}(\mathbf{t}) + \left\langle \mathbf{t}, \nabla(\nabla\psi)^{-1}(\mathbf{t}) \right\rangle - \left\langle \nabla(\nabla\psi)^{-1}(\mathbf{t}), \mathbf{t} \right\rangle \\
&= (\nabla\psi)^{-1}(\mathbf{t})
\end{aligned}$$

□

3. Relation avec les divergences de Bregman

On peut tout d'abord mettre en lumière la relation entre le paramètre naturel $\boldsymbol{\eta}$ d'une famille exponentielle, et le paramètre d'espérance de cette même famille, *i.e.* l'espérance d'une v.a. X distribué selon une loi de probabilité $\mathrm{p}_{(\psi,\boldsymbol{\eta})} \in \mathcal{F}_\psi$ $\boldsymbol{\mu} = \mathbb{E}_{\mathrm{p}_{(\psi,\boldsymbol{\eta})}}[X] = \int_{\mathbf{x}\in\mathbb{R}^d} \mathbf{x}\mathrm{p}_{(\psi,\boldsymbol{\eta})}(\mathbf{x})\,\mathrm{d}\mathbf{x}$. En dérivant l'expression $\int_{\mathbf{x}\in\mathbb{R}^d} \mathrm{p}_{(\psi,\boldsymbol{\eta})}(\mathbf{x})\,\mathrm{d}\mathbf{x} = 1$ selon $\boldsymbol{\eta}$, on obtient

$$\int_{\mathbf{x}\in\mathbb{R}^d} (\mathbf{x} - \nabla\psi(\boldsymbol{\eta}))\mathrm{p}_{(\psi,\boldsymbol{\eta})}(\mathbf{x})\,\mathrm{d}\mathbf{x} = \boldsymbol{\mu} - \nabla\psi(\boldsymbol{\eta}) = 0$$

d'où la relation $\boldsymbol{\mu}_{\boldsymbol{\eta}} = \nabla\psi(\boldsymbol{\eta})$. La relation entre les gradients d'une fonction de Legendre ψ et de son conjugué $\phi = \psi^*$, donnée en proposition I.4, nous permet d'écrire la relation inverse $\boldsymbol{\eta}_{\boldsymbol{\mu}} = (\nabla\psi)^{-1}(\boldsymbol{\mu}) = \nabla\phi(\boldsymbol{\mu})$. On a donc $\psi^*(\boldsymbol{\mu}) = \phi(\boldsymbol{\mu}) = \langle \boldsymbol{\mu}, \boldsymbol{\eta} \rangle - \psi(\boldsymbol{\eta})$, et l'expression de $\mathrm{p}_{(\psi,\boldsymbol{\eta})}$ en (I.6) devient

$$\begin{aligned}
\mathrm{p}_{(\psi,\boldsymbol{\eta})}(\mathbf{x}) &= \exp(\langle \mathbf{x}, \boldsymbol{\eta} \rangle - \psi(\boldsymbol{\eta}))a(\mathbf{x}) \\
&= \exp(\langle \boldsymbol{\mu}, \boldsymbol{\eta} \rangle - \psi(\boldsymbol{\eta}) + \langle \mathbf{x} - \boldsymbol{\mu}, \boldsymbol{\eta} \rangle)a(\mathbf{x}) \\
&= \exp(\phi(\boldsymbol{\mu}) + \langle \mathbf{x} - \boldsymbol{\mu}, \nabla\phi(\boldsymbol{\mu}) \rangle)a(\mathbf{x}) \\
&= \exp(-d_\phi(\mathbf{x}, \boldsymbol{\mu}) + \phi(\mathbf{x}))a(\mathbf{x}) \\
&= \exp(-d_\phi(\mathbf{x}, \boldsymbol{\mu}))b_\phi(\mathbf{x})
\end{aligned} \tag{I.9}$$

où $b_\phi(\mathbf{x}) = \mathrm{e}^{\phi(\mathbf{x})} a(\mathbf{x})$.

La densité de probabilité d'une loi exponentielle suit donc les mêmes variations qu'une divergence de Bregman définie. Plus précisément la log-vraisemblance de l'élément $\mathbf{x}$ distribué selon $\mathrm{p}_{(\psi,\boldsymbol{\eta})}$ peut s'écrire comme la somme d'une divergence de Bregman et d'une fonction ne dépendant que de $\mathbf{x}$.

On a donc mis en évidence la relation entre famille exponentielle et divergence de Bregman. À toute famille exponentielle $\mathcal{F}_\psi$ correspond une divergence d_ϕ déterminée par la relation $\phi = \psi^*$. De plus le paramètre naturel $\boldsymbol{\eta}$ et l'espérance $\boldsymbol{\mu}$ sont déterminés de manière unique. Une démonstration mathématique formelle que cette relation est en fait une bijection, peut être trouvée dans (Banerjee, Merugu et al. 2005). Le tableau I.2 montre les divergences de Bregman associées aux familles exponentielles usuelles.

Loi	$\mathrm{p}_{(\psi,\boldsymbol{\eta})}(\mathbf{x})$	$\boldsymbol{\eta}$	$\psi(\boldsymbol{\eta})$	$\boldsymbol{\mu}$	$\phi(\boldsymbol{\mu})$	$d_\phi(\mathbf{x},\boldsymbol{\mu})$
$\mathcal{N}(m,\sigma^2)^*$	$\frac{1}{\sqrt{2\pi\sigma^2}}\mathrm{e}^{-\frac{(x-m)^2}{2\sigma^2}}$	$\frac{m}{\sigma^2}$	$\frac{m^2}{2\sigma^2} = \frac{\sigma^2}{2}\boldsymbol{\eta}^2$	m	$\frac{1}{2\sigma^2}\boldsymbol{\mu}^2$	$\frac{1}{2\sigma^2}(x-\boldsymbol{\mu})^2$
$\mathcal{B}(p)$	$p^x(1-p)^{1-x}$	$\ln\left(\frac{p}{1-p}\right)$	$\ln\left(1+\mathrm{e}^{\boldsymbol{\eta}}\right)$	p	$\boldsymbol{\mu}\ln(\boldsymbol{\mu})+(1-\boldsymbol{\mu})\ln(1-\boldsymbol{\mu})$	$x\ln\left(\frac{x}{\boldsymbol{\mu}}\right)+(1-x)\ln\left(\frac{1-x}{1-\boldsymbol{\mu}}\right)$
$\mathcal{P}(\lambda)$	$\frac{\lambda^x \mathrm{e}^{-\lambda}}{x!}$	$\ln(\lambda)$	$\lambda = \mathrm{e}^{\boldsymbol{\eta}}$	λ	$\boldsymbol{\mu}\ln(\boldsymbol{\mu}) - \boldsymbol{\mu}$	$x\ln\left(\frac{x}{\boldsymbol{\mu}}\right) - (x-\boldsymbol{\mu})$
$\mathcal{E}(\lambda)$	$\lambda \mathrm{e}^{-\lambda x}$	$-\lambda$	$-\ln(\lambda) = -\ln(-\boldsymbol{\eta})$	$\frac{1}{\lambda}$	$-\ln(\boldsymbol{\mu}) - 1$	$\frac{x}{\boldsymbol{\mu}} - \ln\left(\frac{x}{\boldsymbol{\mu}}\right) - 1$
$\mathcal{N}(\mathbf{m},\boldsymbol{\Sigma})^*$	$\frac{1}{\sqrt{(2\pi^d)\lvert\boldsymbol{\Sigma}\rvert}}\mathrm{e}^{-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{m})}$	$\boldsymbol{\Sigma}^{-1}\mathbf{m}$	$\frac{1}{2}\mathbf{m}^T\boldsymbol{\Sigma}^{-1}\mathbf{m} = \frac{1}{2}\boldsymbol{\eta}^T\boldsymbol{\Sigma}\boldsymbol{\eta}$	$\mathbf{m}$	$\frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$	$\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$

Table I.2 – Famille exponentielles usuelles et divergences de Bregman

*σ^2 et $\boldsymbol{\Sigma}$ sont considérés constants ici

La loi normale est donc associée à la distance Euclidienne, la loi exponentielle à la divergence de Itakura-Saito, et la loi binomiale à la divergence de Kullblack-Leibler entre les distributions $[x, 1-x]$ et $[p, 1-p]$ sur l'espace $\{0, 1\}$. On peut détailler plus précisément le calcul des paramètres avec l'exemple de la distribution multinomiale.

Exemple I.4

Pour N tirages indépendants, aboutissant chacun au succès d'une des d différentes catégories définie chacune par une probabilité de réussite p_i, la loi multinomiale définit la probabilité d'obtenir la combinaison $\{r_i\}_{i=1}^d$, où r_i est le nombre de succès pour la catégorie i. La loi multinomiale est donc définie par les paramètres N, $\mathbf{p} = \{p_i\}_{i=1}^d$ et sa densité de probabilité est $\mathrm{p}(\mathbf{r}|\mathbf{p}, N) = \frac{N!}{\prod_{i=1}^d r_i!}\prod_{i=1}^d p_i^{r_i}$, pour un vecteur $\mathbf{r} \in \mathbb{Z}^+, \sum_{i=1}^d x_i = N$. Cette dernière condition nous permet d'écrire cette densité comme celle d'une famille exponentielle avec $\mathbf{x} = [r_1, \ldots, r_{d-1}]$ avec le paramètre naturel $\boldsymbol{\eta} = [\ln(\frac{p_1}{p_d}), \ldots, \ln(\frac{p_{d-1}}{p_d})]$ puisque $x_d = N - \sum_{i=1}^{d-1}$.

$$
\begin{aligned}
\mathrm{p}(\mathbf{x}|\mathbf{p}, N) &= \frac{N!}{\prod_{i=1}^d r_i!}\prod_{i=1}^d p_i^{r_i} = \frac{N!}{\prod_{i=1}^d r_i!}\exp\left(\sum_{i=1}^{d-1} r_i\ln(p_i) + r_d\ln(p_d)\right) \\
&= \exp\left(\sum_{i=1}^{d-1} x_i\ln(p_i) + (N - \sum_{i=1}^{d-1} x_i)\ln(p_d)\right) a(\mathbf{x}) \\
&= \exp\left(\sum_{i=1}^{d-1} x_i\ln\left(\frac{p_i}{p_d}\right) + N\ln(p_d)\right) a(\mathbf{x}) \\
&= \exp\left(\langle\mathbf{x}, \boldsymbol{\eta}\rangle + N\ln(p_d)\right) a(\mathbf{x})
\end{aligned}
$$

On a alors $\psi(\boldsymbol{\eta}) = -N\ln(p_d) = N\ln(\frac{1}{p_d}) = N\ln(1 + \frac{1-p_d}{p_d}) = N\ln(1 + \sum_{i=1}^{d-1}\frac{p_i}{p_d}) = N\ln(1 + \sum_{i=1}^{d-1}\mathrm{e}^{\boldsymbol{\eta}_i})$. L'espérance se calcule en utilisant la propriété

$$
\boldsymbol{\mu} = \nabla\psi(\boldsymbol{\eta}) = \left[\frac{N\mathrm{e}^{\boldsymbol{\eta}_i}}{1 + \sum_{j=1}^{d-1}\mathrm{e}^{\boldsymbol{\eta}_j}}\right]_{i=1}^{d-1} = [Np_i]_{i=1}^{d-1}
$$

Puis

$$\phi(\boldsymbol{\mu}) = \langle \boldsymbol{\mu}, \boldsymbol{\eta} \rangle - \psi(\boldsymbol{\eta}) = \sum_{i=1}^{d-1} N p_i \ln(\frac{p_i}{p_d}) + N \ln(p_d) = N \sum_{i=1}^{d-1} p_i \ln(p_i) - N(1-p_d)\ln(p_d) + N\ln(p_d)$$
$$= N \sum_{i=1}^{d} p_i \ln(p_i) = \sum_{i=1}^{d} \boldsymbol{\mu}_i \ln(\frac{\boldsymbol{\mu}_i}{N})$$

Où on définit $\boldsymbol{\mu}_d = N p_d = N - \sum_{i=1}^{d-1} \boldsymbol{\mu}_i$. On trouve alors de manière similaire à l'exemple I.2, que la divergence associée à ϕ est la divergence de Kullblack-Leibler

$$\begin{aligned} d_\phi(\mathbf{x}, \boldsymbol{\mu}) &= \sum_{i=1}^{d} x_i \ln(\frac{x_i}{N}) - \sum_{i=1}^{d} \boldsymbol{\mu}_i \ln(\frac{\boldsymbol{\mu}_i}{N}) - \sum_{i=1}^{d} (x_i - \boldsymbol{\mu}_i)(\ln \frac{\boldsymbol{\mu}_i}{N} + 1) \\ &= \sum_{i=1}^{d} x_i \ln \frac{x_i/N}{\boldsymbol{\mu}_i/N} + \sum_{i=1}^{d} x_i - \boldsymbol{\mu}_i \\ &= N \sum_{i=1}^{d} \frac{x_i}{N} \ln \frac{x_i/N}{\boldsymbol{\mu}_i/N} \quad \text{car} \quad \sum_{i=1}^{d} x_i = \sum_{i=1}^{d} \boldsymbol{\mu}_i = N \end{aligned}$$

Cette relation permet d'apporter une conclusion importante sur le choix d'une métrique ou d'un modèle probabiliste : en effet, utiliser une certaine divergence de Bregman comme mesure de similarité entre un centroïde et un élément dans un ensemble donné revient à considérer que les éléments sont répartis selon une loi de famille exponentielle, dont la fonction de log-partition et le paramètre naturel sont déterminés de manière unique par la fonction de Bregman convexe et le centroïde d'après la relation (I.9). De manière équivalente, si l'on considère un modèle de mélange suivant une loi exponentielle donnée, cela implique l'utilisation de la divergence de Bregman correspondante comme métrique.

On remarquera que dans le cas de la KL-divergence, la loi correspondante est la loi multinomiale, ce qui implique $\mathbf{x} \in \mathbb{N}$ avec $\sum_{x \in \mathbf{x}} x = N$. Dans le cas pratique du partitionnement audio, on pourra normaliser les descripteurs à chaque temps t, multiplier par une constante N suffisamment grande et discrétiser pour approximer ce comportement. On verra que, la fonction b_ϕ n'ayant pas besoin d'être calculée pour nos algorithmes, il n'est en fait pas nécessaire d'avoir des descripteurs discrets dans ce cas.

4. Modèle de mélange et partitionnement

1. Modèle de mélange

Soit un ensemble de points $\mathcal{X}$ que l'on cherche à partitionner. On parle de modèle de mélange lorsque l'on considère les données à classifier comme des observations émises selon des variables latentes définissant la classe de l'observation. Chaque composante émet des observations selon une loi de probabilité définie par sa classe, ces composantes suivant une loi discrète définissant la proportion du mélange. Formellement, pour un partitionnement en K classes, on définit pour chaque élément une variable latente h à valeur dans $[\![1; K]\!]$ déterminant sa classe. On a $\mathrm{p}(h = k) = \pi_k$, où $\boldsymbol{\pi}$ est la probabilité discrète définissant la répartition du mélange. On définit pour chaque classe une loi d'émission de densité p_k et de paramètre $\boldsymbol{\alpha}_k$, tel que $\mathrm{p}(\mathbf{x}|h = k) = \mathrm{p}_k(\mathbf{x}; \boldsymbol{\alpha}_k)$. Tout vecteur suivra

donc la loi de mélange définie par $\boldsymbol{\theta} = (\pi_1, \ldots, \pi_K, \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_K)$, de densité

$$\mathrm{p}(\mathbf{x}; \boldsymbol{\theta}) = \sum_{k=1}^{K} \mathrm{p}(\mathbf{x}|h = k; \boldsymbol{\theta})\mathrm{p}(h = k; \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathrm{p}_k(\mathbf{x}; \boldsymbol{\alpha}_k) \tag{I.10}$$

Si on considère l'ensemble des probabilités d'observation comme appartenant à une même famille exponentielle $\mathcal{F}_\psi$, chacune définie par un paramètre naturel $\boldsymbol{\eta}_k$ propre, la densité de mélange devient, en considérant la bijection avec l'ensemble des divergences de Bregman, $\mathrm{p}_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathrm{e}^{-d_\phi(\mathbf{x}, \boldsymbol{\mu}_k)} b_\phi(\mathbf{x})$. La figure I.1 montre un exemple de densité de mélange dans le cas de densités gaussiennes, pour un espace à une dimension.

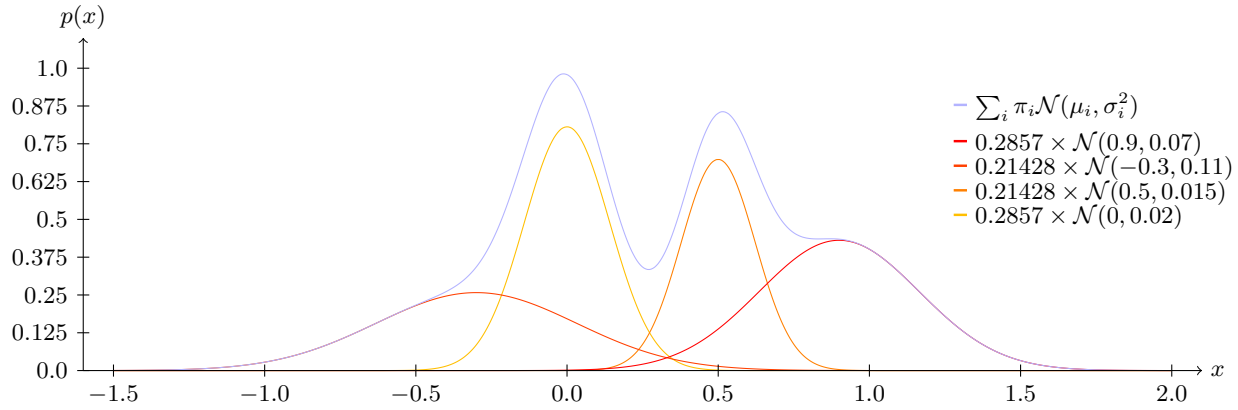


FIGURE I.1 – Un exemple de modèle de mélange gaussien

Le but d'un algorithme de classification basé sur un tel modèle sera de maximiser la vraisemblance d'un jeu de données, selon la densité définie par (I.10), ou sa log-vraisemblance. Pour un ensemble $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, on va donc chercher à faire croître la fonction $l(\boldsymbol{\theta}) = \ln(\mathrm{p}(\mathcal{X}; \boldsymbol{\theta})) = \sum_{i=1}^{n} \ln(\sum_{k=1}^{K} \pi_k \mathrm{p}_k(\mathbf{x}_i; \boldsymbol{\alpha}_k))$, selon le paramètre $\boldsymbol{\theta}$. Contrairement au problème de minimisation de la fonction de perte (I.5) dans le cas du partitionnement dur, la fonction de log-vraisemblance n'est en général pas convexe. Il est donc impossible de trouver un maximum global, on peut en revanche en trouver un maximum local, c'est le rôle d'un algorithme permettant d'estimer le maximum de vraisemblance de données incomplètes, tel que l'algorithme EM.

2. Espérance-Maximisation

L'algorithme Espérance-Maximisation (EM, *Expectation-Maximization*) est un algorithme permettant selon un jeu de données observables, et un jeu de données latentes (dans le cas du partitionnement, la classe à laquelle appartient chaque donnée) d'estimer le maximum de vraisemblance pour un modèle donné, et d'en inférer les données non observables (DEMPSTER et al. 1977). Tout comme pour l'algorithme des K-Moyennes, il fonctionne de manière itératif en deux étapes : le calcul de l'espérance des variables latentes selon les observations, puis le calcul des paramètres maximisant la vraisemblance déduite.

Théorie

On présente ici les outils formels sur lesquels s'appuient l'algorithme, et la preuve de sa convergence (DELLAERT 2002). L'objectif est donc de trouver

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) \tag{I.11}$$

L'idée est de partir d'un paramètre initial $\boldsymbol{\theta}^{(0)}$, trouver la meilleure limite inférieure à $l(\boldsymbol{\theta})$, et itérer sur t pour trouver $\boldsymbol{\theta}^{(\tau)}$ maximisant cette limite inférieure. On note $B(\boldsymbol{\theta}, \boldsymbol{\theta}^{(\tau)})$ la borne définie par $\boldsymbol{\theta}^{(\tau)}$. On peut remarquer que, d'après l'inégalité de Jensen $f(\sum_i \lambda_i \mathbf{x}_i) \geq \sum_i \lambda_i f(\mathbf{x}_i)$ pour f convexe et $\sum_i \lambda_i = 1$, on a

$$l(\boldsymbol{\theta}) = \ln(\mathrm{p}(\mathcal{X}; \boldsymbol{\theta})) = \ln(\sum_{\mathbf{h}} \mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta})) \geq \sum_{\mathbf{h}} q^{(\tau)}(\mathbf{h}) \ln(\frac{\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta})}{q^{(\tau)}(\mathbf{h})}) = B(\boldsymbol{\theta}, \boldsymbol{\theta}^{(\tau)})$$

où $q^{(\tau)}$ est une fonction à définir telle que $\sum_{\mathbf{h}} q^{(\tau)}(\mathbf{h}) = 1$. On cherche à trouver $q^{(\tau)}$ tel que cette limite inférieure soit optimale, cela se fait en maximisant $B(\boldsymbol{\theta}^{(\tau)}, \boldsymbol{\theta}^{(\tau)})$ selon $q^{(\tau)}$ en introduisant un multiplicateur de Lagrange λ pour respecter la condition $\sum_{\mathbf{h}} q^{(\tau)}(\mathbf{h}) = 1$. On dérive ensuite la fonction objectif :

$$G(q^{(\tau)}) = \lambda \left[1 - \sum_{\mathbf{h}} q^{(\tau)}(\mathbf{h})\right] + \sum_{\mathbf{h}} q^{(\tau)}(\mathbf{h}) \ln(\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta}^t)) - \sum_{\mathbf{h}} q^{(\tau)}(\mathbf{h}) \ln(q^{(\tau)}(\mathbf{h}))$$

$$\frac{\delta G}{\delta q^{(\tau)}(\mathbf{h})}(q^{(\tau)}) = -\lambda + \ln(\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta})) - \ln(q^{(\tau)}(\mathbf{h})) - 1 = 0$$

En sommant les exponentielles de la deuxième expression pour tout $\mathbf{h}$, on trouve $\lambda + 1 = \ln(\sum_{\mathbf{h}} \mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta}))$. On en conclut donc

$$q^{(\tau)}(\mathbf{h}) = \frac{\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta}^{(\tau)})}{\sum_{\mathbf{h}} \mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta}^{(\tau)})} = \frac{\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta}^{(\tau)})}{\mathrm{p}(\mathcal{X}; \boldsymbol{\theta}^{(\tau)})} = \mathrm{p}(\mathbf{h}|\mathcal{X}; \boldsymbol{\theta}^{(\tau)}) \tag{I.12}$$

On trouve que la limite atteint bien la fonction objectif $l(\boldsymbol{\theta}^{(\tau)})$ avec $\boldsymbol{\theta} = \boldsymbol{\theta}^{(\tau)}$.

$$B(\boldsymbol{\theta}^{(\tau)}, \boldsymbol{\theta}^{(\tau)}) = \sum_{\mathbf{h}} \mathrm{p}(\mathbf{h}|\mathcal{X}; \boldsymbol{\theta}^{(\tau)}) \ln\left(\frac{\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta}^{(\tau)})}{\mathrm{p}(\mathbf{h}|\mathcal{X}; \boldsymbol{\theta}^{(\tau)})}\right) = \sum_{\mathbf{h}} \mathrm{p}(\mathbf{h}|\mathcal{X}; \boldsymbol{\theta}^{(\tau)}) \ln(\mathrm{p}(\mathcal{X}; \boldsymbol{\theta}^{(\tau)})) = \ln(\mathrm{p}(\mathcal{X}; \boldsymbol{\theta}^{(\tau)}))$$

Il convient ensuite de trouver le nouveau paramètre $\boldsymbol{\theta} = \boldsymbol{\theta}^{(\tau+1)}$ maximisant cette borne inférieure. On note que

$$B(\boldsymbol{\theta}, \boldsymbol{\theta}^{(\tau)}) = \mathbb{E}\Big[\ln(\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta}))|\mathcal{X}; \boldsymbol{\theta}^{(\tau)}\Big] - \mathbb{E}\Big[\ln(\mathrm{p}(\mathbf{h}|\mathcal{X}; \boldsymbol{\theta}^{(\tau)}))|\mathcal{X}; \boldsymbol{\theta}^{(\tau)}\Big]$$

Seule la première espérance dépend de $\boldsymbol{\theta}$, on peut donc en déduire qu'il de suffit maximiser cette espérance. Les deux étapes de l'algorithme en découlent :

Espérance Calcul de $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(\tau)}) = \mathbb{E}\Big[\ln(\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta}))|\mathcal{X}; \boldsymbol{\theta}^{(\tau)}\Big]$

Maximisation Calcul de $\boldsymbol{\theta}^{(\tau+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(\tau)})$

La fonction de log-vraisemblance $l(\boldsymbol{\theta}^{(\tau)})$ va donc croître, et converger vers un maximum local étape par étape, puisque la borne inférieure va augmenter. On peut en général arrêter l'algorithme lorsque la différence de la log-vraisemblance entre deux itération devient relativement faible. Puisque le maxium vers lequel converge l'algorithme est local, la solution dépend fortement de l'initialisation de $\boldsymbol{\theta}^{(\tau)}$.

Cas des familles exponentielles

On considère que notre modèle est un mélange de distribution d'une famille exponentielle $\mathcal{F}_\psi$ de paramètres $\boldsymbol{\eta}_k, k \in [\![1; K]\!]$. Écrivons tout d'abord la loi jointe $\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta})$ afin de maximiser son espérance.

$$\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta}) = \prod_{i=1}^{n} \mathrm{p}(\mathbf{x}_i|h_i; \boldsymbol{\eta}_{h_i}) \mathrm{p}(h_i; \pi_{h_i}) = \prod_{i=1}^{n} \mathrm{p}_{(\psi, \boldsymbol{\eta}_{h_i})}(\mathbf{x}_i) \pi_{h_i} \tag{I.13}$$

Pour trouver le paramètre $\boldsymbol{\eta}_k^{(\tau+1)}$ maximisant $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(\tau)})$ on maximise la contribution de la famille exponentielle k, *i.e.*

$$\boldsymbol{\eta}_k^{(\tau+1)} = \arg\max_{\boldsymbol{\eta}_k} \sum_{i=1}^{n} \mathrm{p}(h_i = k|\mathbf{x}_i; \boldsymbol{\theta}^{(\tau)}) \ln(\mathrm{p}_{(\psi,\boldsymbol{\eta}_k)}(\mathbf{x}_i))$$

Puisque la famille est exponentielle, on peut expliciter la densité en fonction de la divergence de Bregman associée (I.9), et comme $\boldsymbol{\eta}$ et $\boldsymbol{\mu}$ sont déterminée de manière unique, on peut en déduire $\boldsymbol{\mu}_k^{(\tau)}$ d'après l'expression de la densité de la famille exponentielle (I.6)

$$\begin{aligned}
\boldsymbol{\mu}_k^{(\tau+1)} &= \arg\min_{\boldsymbol{\mu}_k} \sum_{i=1}^{n} \mathrm{p}(h_i = k|\mathbf{x}_i; \boldsymbol{\theta}^{(\tau)}) d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_k) \\
&= \arg\min_{\boldsymbol{\mu}_k} \sum_{i=1}^{n} d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_k) \frac{\mathrm{p}(h_i = k|\mathbf{x}_i; \boldsymbol{\theta}^{(\tau)})}{\sum_{j=1}^{n} \mathrm{p}(h_i = k|\mathbf{x}_i; \boldsymbol{\theta}^{(\tau)})}
\end{aligned}$$

D'après la proposition I.1, le $\boldsymbol{\mu}_k^{(\tau+1)}$ vérifiant ceci est l'espérance de $\mathbf{x}$ selon la mesure définie. Donc en notant $\tau_i^{(\tau)}(k) = \mathrm{p}(h_i = k|\mathbf{x}_i; \boldsymbol{\theta}^{(\tau)})$,

$$\boldsymbol{\mu}_k^{(\tau+1)} = \frac{\sum_{i=1}^{n} \tau_i^{(\tau)}(k)\mathbf{x}_i}{\sum_{i=1}^{n} \tau_i^{(\tau)}(k)} \tag{I.14}$$

Le paramètre $\boldsymbol{\pi}^{(\tau+1)}$ doit de même maximiser $\sum_{i=1}^{n}\sum_{k=1}^{K} \tau_i^{(\tau)}(k) \ln(\pi_k)$. On utilise alors le Lagrangien

$$\begin{aligned}
G(\boldsymbol{\pi}) &= \sum_{i=1}^{n}\sum_{k=1}^{K} \tau_i^{(\tau)}(k) \ln(\pi_k) + \lambda \left[1 - \sum_{k=1}^{K} \pi_k\right] \\
\frac{\delta G}{\delta \pi_k}(\boldsymbol{\pi}) &= \sum_{i=1}^{n} \tau_i^{(\tau)}(k) \ln(\pi_k) - \lambda = 0
\end{aligned}$$

En sommant la deuxième expression pour tout k, on obtient $\lambda = \sum_{i=1}^{n}\sum_{k=1}^{K} \tau_i^{(\tau)}(k) = n$ d'où

$$\pi_k^{(\tau+1)} = \frac{1}{n} \sum_{i=1}^{n} \tau_i^{(\tau)}(k) \tag{I.15}$$

On peut expliciter la densité *a priori* $\tau_i^{(\tau)}(k)$ en utilisant la densité de la famille exponentielle.

$$\begin{aligned}
\tau_i^{(\tau)}(k) &= \frac{\mathrm{p}(h_i = k, \mathbf{x}_i; \boldsymbol{\theta}^{(\tau)})}{\mathrm{p}(\mathbf{x}_i; \boldsymbol{\theta}^{(\tau)})} \\
&= \frac{\mathrm{p}(h_i = k, \mathbf{x}_i; \boldsymbol{\theta}^{(\tau)})}{\sum_{k=1}^{K} \mathrm{p}(h_i = k, \mathbf{x}_i; \boldsymbol{\theta}^{(\tau)})} \\
&= \frac{\pi_k \mathrm{e}^{-d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_k^{(\tau)})}}{\sum_{j=1}^{K} \pi_j \mathrm{e}^{-d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_j^{(\tau)})}}
\end{aligned} \tag{I.16}$$

Ce résultat est important, en effet on n'a pas besoin de calculer la densité de probabilité pour connaître la densité *a priori*, il suffit de connaitre la divergence de Bregman associée. Cela est notamment utile lorsqu'on utilise un modèle de mélange multinomial, le terme combinatoire $N!/\prod_i x_i!$ disparaît, il suffit d'utiliser la divergence de Kullblack-Leibler. De même tout terme constant selon $\mathbf{x}$ dans $d_\phi(\mathbf{x}, \boldsymbol{\mu})$ peut être enlevé (par exemple $\sum_j x_j \ln(x_j)$).

Algorithme 2 Espérance-Maximisation

Procédure ESPERANCE-MAXIMISATION($\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \Omega$, $\mathcal{F}_\psi$,K)
  Initialise $\{\boldsymbol{\mu}_k\}_{k=1}^K$, généralement par K-Moyennes, et $\{\pi_k\}_{k=1}^K$, généralement $\pi_k = \frac{1}{K}$.
  Répète pour $t \in \mathbb{N}$
    Pour $i \leftarrow 1, \dots, n$ **faire**
      Pour $k \leftarrow 1, \dots, K$ **faire**
        $\tau_i^{(\tau)}(k) \leftarrow \pi_k \mathrm{e}^{-d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_k^{(\tau)})}$
      Fin Pour
      $\tau_i^{(\tau)}(k) \leftarrow \frac{\tau_i^{(\tau)}(k)}{\sum_{j=1}^K \tau_i^{(\tau)}(k)}$
    Fin Pour
    Pour $k \leftarrow 1, \dots, K$ **faire**
      $\pi_k^{(\tau+1)} \leftarrow \frac{1}{n} \sum_{i=1}^n \tau_i^{(\tau)}(k)$
      $\boldsymbol{\mu}_k^{(\tau+1)} \leftarrow \frac{\sum_{i=1}^n \tau_i^{(\tau)}(k) \mathbf{x}_i}{\sum_{i=1}^n \tau_i^{(\tau)}(k)}$
    Fin Pour
  Jusqu'à convergence
  Retourne $\{\pi_k\}_{k=1}^K, \{\boldsymbol{\mu}_k\}_{k=1}^K, \{\tau_{i,k}\}_{i,k=1}^{n,K}$
Fin Procédure

Algorithme

Le déroulement de l'algorithme découle immédiatement des équations (I.14),(I.15),(I.16). Il est décrit par l'algorithme 2.

À l'issue de l'algorithme, on peut déterminer un partitionnement de $\mathcal{X}$ en prenant $h_i = \arg\max_k \tau_i(k)$. On comprend pourquoi le partitionnement est qualifié de *soft clustering*, puisqu'à chaque itération, un élément n'est pas assigné à une classe. On calcule plutôt la probabilité de chaque élément d'appartenir à chaque classe selon le modèle.

5. Modèles de Markov Caché

Le modèle de mélange décrit précédemment considère que les observations sont indépendantes et identiquement distribuées, cela n'est bien sûr pas du tout le cas dans le cas de signaux audio. Les différentes observations contenues dans les descripteurs sont dépendantes en temps. Pour prendre en compte la dimension temporelle, indispensable en apprentissage statistique appliqué à l'audio, on utilise les modèles de Markov cachées (*Hidden Markov Models*, HMM) (L.R. RABINER et al. 1989 ; SMYTH 1997).

1. Modélisation

Tout comme pour le modèle de mélange, on considère que les données $\mathcal{X} = \{\mathbf{x}_t\}_{t=1}^T$ sont des observations émises pour des variables cachées $\{h_t\}_{t=1}^T$, cependant, on considère que la séquence des variables cachées suit une chaîne de Markov. On rappelle qu'une chaîne de Markov est un processus d'états à temps discret respectant la propriété de Markov : la prédiction de l'état futur ne dépend que de l'état présent. Pour une séquence de variables latentes $[h_{1:T}]$, on a pour tout instant t

$$\mathrm{p}(h_{t+1} = j | h_t = i, h_{t-1:1}) = \mathrm{p}(h_{t+1} = j | h_t = i) = a_{i,j}$$

où $a_{i,j}$ est un élément d'une matrice $\mathbf{A} = (a_{i,j})_{i,j \in [\![1;K]\!]}$ telle que $\sum_{j=1}^K a_{i,j} = 1 \quad \forall i \in [\![1;K]\!]$. On considère aussi une probabilité à l'instant initial $\pi_i = \mathrm{p}(h_1 = i)$ pour les variables cachées. De

même que précédemment on modélisera l'émission des observations par une famille exponentielle $\mathcal{F}_\psi$ où chaque classe possède un paramètre naturel $\boldsymbol{\eta}$ propre, et donc une densité de probabilité $\mathrm{p}_{\psi,\boldsymbol{\eta}}(\mathbf{x}) = \mathrm{p}(\mathbf{x}|h)$ et une divergence de Bregman $d_\phi(\mathbf{x}, \boldsymbol{\mu})$ propres. Les paramètres du modèle pour K états différents sont alors $\boldsymbol{\theta} = (\mathbf{A}, \boldsymbol{\pi}, \{\boldsymbol{\mu}_k\}_{k=1}^K)$.

La figure I.2 représente la modélisation d'un HMM sous forme de graphe, où (a) représente la séquence des états selon la valeur des variables cachées, ainsi que les émissions d'observations, selon les différentes densités de probabilité définies ; tandis que (b) représente une chaîne de Markov à trois états sous forme d'un graphe orienté et pondéré par les probabilités de transition.

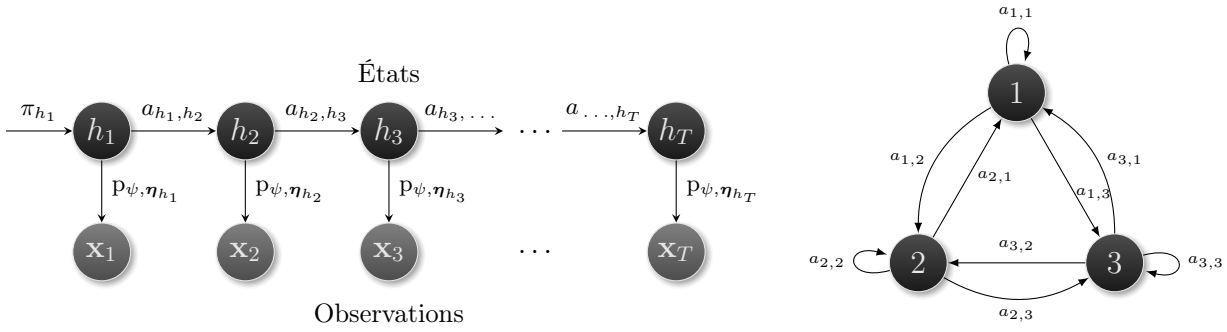


FIGURE I.2 – Modélisation par graphe de HMM

2. Décodage et évaluation

Algorithme de Viterbi

Le premier problème posé par le modèle, et qui fait partie des trois problèmes classiques des HMM, est de connaître la séquence d'état la plus probable selon un modèle défini par $\boldsymbol{\theta}$, étant donné une séquence d'observation. Formellement on cherche $\mathbf{h}^* = \arg\max_{\mathbf{h}} \mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta})$. Ce problème est résolu par l'algorithme de Viterbi. L'idée est de définir une variable intermédiaire $\delta_t(i)$ définie comme la probabilité maximale d'observer les t premières observation ainsi qu'une séquence d'état arrivant à l'état i en t :

$$\delta_t(i) = \max_{h_{1:t}} \mathrm{p}(\mathbf{x}_{1:t}, h_{1:t-1}, h_t = i; \boldsymbol{\theta}) \tag{I.17}$$

Étant donné la nature sans mémoire des chaînes de Markov, et que les observations ne dépendent que de la variable cachée correspondante, on peut facilement décomposer cette probabilité en :

$$\begin{aligned}
&\mathrm{p}(\mathbf{x}_t|h_{1:t}, \mathbf{x}_{1:t-1}; \boldsymbol{\theta})\mathrm{p}(h_t|h_{1:t-1}, \mathbf{x}_{1:t-1}; \boldsymbol{\theta})\mathrm{p}(h_{1:t-1}, \mathbf{x}_{1:t-1}; \boldsymbol{\theta}) \\
=&\mathrm{p}(\mathbf{x}_t|h_t)\mathrm{p}(h_t|h_{t-1})\mathrm{p}(h_{1:t-1}, \mathbf{x}_{1:t-1}; \boldsymbol{\theta}) \\
=&\mathrm{p}_{(\psi,\boldsymbol{\eta}_{h_t})}(\mathbf{x}_t)a_{h_{t-1},h_t}\mathrm{p}(h_{1:t-1}, \mathbf{x}_{1:t-1}; \boldsymbol{\theta})
\end{aligned}$$

On en conclut que $\delta_t(i)$ peut s'écrire de manière récursive :

$$\delta_t(j) = \mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_t) \max_{i \in [\![1;K]\!]} a_{i,j}\delta_{t-1}(i) \tag{I.18}$$

Avec $\delta_1(i) = \mathrm{p}(\psi, \boldsymbol{\eta}_i)(\mathbf{x}_1)\pi_i$. En calculant récursivement $\delta_t(j)$ pour tout j de 1 à T, on peut finalement déterminer quel est l'état le plus probable à la fin de la séquence en calculant $h_T^* = \arg\max_j \delta_T(j)$. En définissant des pointeurs arrières

$$\gamma_t(j) = \arg\max_{i \in [\![1;K]\!]} a_{i,j}\delta_{t-1}(i) \tag{I.19}$$

on peut retrouver la séquence la plus probable avec $h^*_{t-1} = \gamma_t(h^*_t)$ pour $t \in [\![2; T]\!]$.

L'algorithme de Viterbi est schématisé par la figure I.3, les flèches grises entrant dans un état représentent le calcul de $\delta_t(j)$ défini par (I.18). Les flèches orange la séquence la plus probable retrouvée grâce à $\gamma_t(j)$ (I.19).

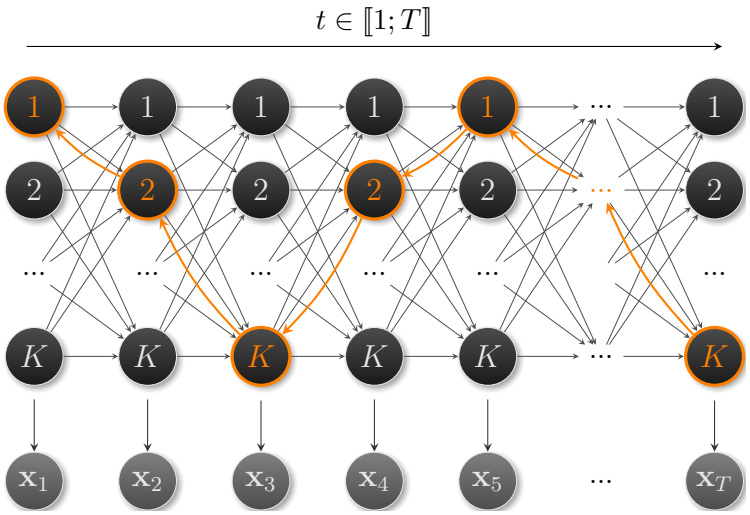
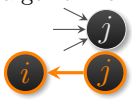


FIGURE I.3 – Algorithme de Viterbi

L'algorithme fonctionne en deux étapes :



Calcul récursif de $\delta_t(j)$ depuis $\delta_{t-1}(i)$ $\forall i \in [\![1, K]\!]$ et des paramètre $\boldsymbol{\theta}$.

Découverte de la séquence optimale en partant de $\arg\max_j \delta_T(j)$ grâce à $\gamma_t(j)$.

Algorithmes *forward-backward*

Le second problème posé par les HMM est, étant donné un modèle définit par $\boldsymbol{\theta}$, de déterminer la probabilité d'une séquence d'observations $\mathrm{p}(\mathbf{x}_{1:T}; \boldsymbol{\theta})$. Une solution possible sera de générer l'ensemble de séquences d'états possibles et de calculer $\sum_{\mathbf{h}} \mathrm{p}(\mathbf{x}_{1:T}|\mathbf{h}; \boldsymbol{\theta})\mathrm{p}(\mathbf{h}; \boldsymbol{\theta})$ mais cela implique un nombre exponentiel de probabilités à calculer. On définit plutôt $\alpha_t(j) = \mathrm{p}(\mathbf{x}_{1:t}, h_t = j; \boldsymbol{\theta})$. On peut alors redéfinir la probabilité de la séquence d'observations comme la somme $\sum_i \alpha_T(j)$, et de la même manière que pour l'algorithme de Viterbi, on peut calculer $\alpha_t(j)$ pour tout j et t de manière récurrente :

$$\begin{aligned}
\alpha_t(j) &= \sum_{i=1}^{K} \mathrm{p}(\mathbf{x}_{1:t}, h_{t-1} = i, h_t = j; \boldsymbol{\theta}) \\
&= \sum_{i=1}^{K} \mathrm{p}(\mathbf{x}_t|h_t = j; \boldsymbol{\theta})\mathrm{p}(h_t = j|h_{t-1} = i; \boldsymbol{\theta})\mathrm{p}(\mathbf{x}_{1:t-1}, h_{t-1} = i; \boldsymbol{\theta}) \\
&= \mathrm{p}_{(\psi, \boldsymbol{\eta}_j)}(\mathbf{x}_t) \sum_{i=1}^{K} a_{i,j}\alpha_{t-1}(i)
\end{aligned} \tag{I.20}$$

Et on peut initialiser avec $\alpha_1(i) = \mathrm{p}(\mathbf{x}_1|h_1 = i; \boldsymbol{\theta})\mathrm{p}(h_1 = i; \boldsymbol{\theta}) = \mathrm{p}_{(\psi, \boldsymbol{\eta}_i)}(\mathbf{x}_1)\pi_i$. Cette récursion définit l'algorithme *forward*.

On remarque que si on veut calculer l'inférence d'un état sur l'ensemble des données, qui permet tout comme pour l'algorithme EM d'effectuer un apprentissage du modèle comme nous le verrons,

on va pouvoir utiliser cet algorithme. En effet

$$\begin{aligned}\mathrm{p}(h_t = i|\mathcal{X};\boldsymbol{\theta}) &= \frac{1}{\mathrm{p}(\mathcal{X};\boldsymbol{\theta})}\mathrm{p}(h_t = i,\mathcal{X};\boldsymbol{\theta})\\ \mathrm{p}(h_t = i,\mathcal{X};\boldsymbol{\theta}) &= \mathrm{p}(\mathbf{x}_{t+1:T}|\mathbf{x}_{1:t},h_t = i;\boldsymbol{\theta})\mathrm{p}(\mathbf{x}_{1:t},h_t = i;\boldsymbol{\theta})\\ &= \mathrm{p}(\mathbf{x}_{t+1:T}|h_t = i;\boldsymbol{\theta})\alpha_t(i)\end{aligned}$$

La valeur $\beta_t(i) = \mathrm{p}(\mathbf{x}_{t+1:T}|h_t = i;\boldsymbol{\theta})$ peut être calculée, de même que pour *forward*, de manière récursive et définit l'algorithme *backward*.

$$\begin{aligned}\beta_t(i) &= \sum_{j=1}^{K}\mathrm{p}(\mathbf{x}_{t+1:T},h_{t+1} = j|h_t = i;\boldsymbol{\theta})\\ &= \sum_{j=1}^{K}\mathrm{p}(\mathbf{x}_{t+1}|h_{t+1} = j;\boldsymbol{\theta})\mathrm{p}(h_{t+1} = j|h_t = i;\boldsymbol{\theta})\mathrm{p}(\mathbf{x}_{t+2:T},|h_{t+1} = j;\boldsymbol{\theta})\\ &= \sum_{j=1}^{K}\mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_{t+1})a_{i,j}\beta_{t+1}(j)\end{aligned} \tag{I.21}$$

Avec $\beta_T(i) = 1 \quad \forall i \in [\![1;K]\!]$. On peut facilement calculer le terme d'inférence dit de *smoothing*, en remarquant que $\mathrm{p}(h_t = i,\mathcal{X};\boldsymbol{\theta}) = \alpha_t(i)\beta_t(i)$ et que $\mathrm{p}(\mathcal{X};\boldsymbol{\theta}) = \sum_{i=1}^{n}\alpha_T(i)$ tel qu'expliqué précédemment (on peut remarquer que cette relation est un cas particulier de $\mathrm{p}(\mathcal{X};\boldsymbol{\theta}) = \sum_{i=1}^{K}\alpha_t(i)\beta_t(i)$ pour tout t.)

3. Apprentissage, EM

On va maintenant pouvoir étendre l'algorithme EM au modèle HMM, il suffit pour cela de remarquer que la probabilité jointe des variables cachées et des observations est

$$\begin{aligned}\mathrm{p}(\mathcal{X},\mathbf{h};\mathbf{A},\boldsymbol{\pi},\{\mu_k\}_{k=1}^{K}) &= \mathrm{p}(h_1;\boldsymbol{\pi})\prod_{t=2}^{T}\mathrm{p}(h_t|h_{t-1};\mathbf{A})\prod_{t=1}^{T}\mathrm{p}(\mathbf{x}_t|h_t;\{\boldsymbol{\mu}_k\}_{k=1}^{K})\\ &= \pi_{h_1}\prod_{t=2}^{T}\mathbf{A}_{h_{t-1},h_t}\prod_{t=1}^{T}\mathrm{p}_{(\psi,\boldsymbol{\eta}_{h_t})}(\mathbf{x}_t)\end{aligned}$$

L'algorithme d'apprentissage fonctionne de la même manière que pour EM, on calcule d'abord $\mathbb{E}_{\mathbf{h}|\mathcal{X},\boldsymbol{\theta}^{(\tau)}}[\ln(\mathrm{p}(\mathcal{X},\mathbf{h};\boldsymbol{\theta}))]$, puis on met à jour en sélectionnant les paramètre $\boldsymbol{\theta}^{(\tau+1)}$ maximisant cette fonction. D'après l'expression de la probabilité jointe, on peut mettre à jour les centroïdes $\boldsymbol{\mu}_i^{(\tau+1)}$ de la même manière pour le modèle de mélange (I.14). Il suffit de définir comme pour le modèle de mélange, et d'après ce qu'on a énoncé avec les algorithmes *forward-backward*.

$$\tau_t^{(\tau)}(i) = \frac{\alpha_t^{(\tau)}(i)\beta_t^{(\tau)}(i)}{\sum_{j=1}^{K}\alpha_t^{(\tau)}(j)\beta_t^{(\tau)}(j)} \tag{I.22}$$

Pour la probabilité initiale $\boldsymbol{\pi}$ on met à jour avec

$$\begin{aligned}\boldsymbol{\pi}^{(\tau+1)} &= \arg\max_{\boldsymbol{\pi}}\ \sum_{i=1}^{K}p(h_1 = i|\mathcal{X};\boldsymbol{\theta}^{(\tau)})\ln(\pi_i)\\ &= \arg\max_{\boldsymbol{\pi}}\ \sum_{i=1}^{K}\tau_1^{(\tau)}(i)\ln(\pi_i)\end{aligned} \tag{I.23}$$

La condition $\sum_{i=1}^{K} \pi_i = 1$ donne le résultat $\pi_i^{(\tau+1)} = \tau_1^{(\tau)}(i)$.

Il reste enfin à trouver $\mathbf{A}^{(\tau+1)}$ maximisant l'espérance. On définit tout d'abord la variable $\epsilon_t^{(\tau)}(i,j)$ comme la probabilité d'être à l'état i en t et à l'état j en $t+1$ selon les observations. On remarque en suivant un raisonnement similaire à $\tau_t^{(\tau)}(i)$

$$\begin{aligned} \mathrm{p}(h_t = i, h_{t+1} = j, \mathcal{X}) &= \mathrm{p}(\mathbf{x}_{l+2:T}|h_{t+1} = j)\mathrm{p}(h_{t+1}|h_t = i)\mathrm{p}(\mathbf{x}_{t+1}|h_{t+1} = j)\mathrm{p}(h_t = i, \mathbf{x}_{1:l}) \\ &= \beta_{t+1}(j)a_{i,j}\mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_{t+1})\alpha_t(i) \\ \text{d'où} \quad \epsilon_t^{(\tau)}(i,j) &= \frac{\beta_{t+1}^{(\tau)}(j)a_{i,j}^{(\tau)}\mathrm{p}_{(\psi,\boldsymbol{\eta}_j^{(\tau)})}(\mathbf{x}_{t+1})\alpha_t^{(\tau)}(i)}{\sum_{i'=1}^{K}\sum_{j'=1}^{K}\beta_{t+1}^{(\tau)}(j')\mathrm{p}_{(\psi,\boldsymbol{\eta}_j^{(\tau)})}(\mathbf{x}_{t+1})\alpha_t^{(\tau)}(i')} \end{aligned} \tag{I.24}$$

On cherche alors, pour $(\mathbf{A})_i$ désignant la ligne i de la matrice $\mathbf{A}$, et en utilisant un Lagrangien comme (I.1) p.15, avec $\sum_j a_{i,j} = 1$ et $\sum_t \sum_j \epsilon_t^{(\tau)}(i,j) = \sum_t \tau_t^{(\tau)}(i)$.

$$\begin{aligned} (\mathbf{A})_i^{(\tau+1)} &= \arg\max_{(\mathbf{A})_i} \sum_{t=1}^{T}\sum_{j=1}^{K} \epsilon_t^{(\tau)}(i,j)\ln(a_{i,j}) \\ a_{i,j}^{(\tau+1)} &= \frac{\sum_{t=1}^{T}\epsilon_t^{(\tau)}(i,j)}{\sum_{t=1}^{T}\tau_t^{(\tau)}(i)} \end{aligned} \tag{I.25}$$

L'algorithme d'apprentissage, dit aussi algorithme de Baum-Welch, se déroule donc ainsi en itérant sur t

- Calcul de variables *forward-backward* $\alpha_t^{(\tau)}(i), \beta_t^{(\tau)}(i)$ pour tout $i \in [\![1;K]\!], t \in [\![1,T]\!]$ (I.20), (I.21)
- Calcul de $\tau_t^{(\tau)}(i)$ et $\epsilon_t^{(\tau)}(i,j)$ pour tout $i \in [\![1;K]\!], t \in [\![1,T]\!]$ (I.22),(I.24).
- Maximisation : mise à jour des paramètres $\boldsymbol{\pi}$ (I.23), $\boldsymbol{\mu}$ (I.14) et $\mathbf{A}$ (I.25).

On remarquera que les calculs des densités $\mathrm{p}_{(\psi,\boldsymbol{\eta})}$ ne s'effectuent que lors de l'algorithme *forward-backward*. Étant donné qu'on normalise lors du calcul des probabilités *a priori* $\tau_t^{(\tau)}(i), \epsilon_t^{(\tau)}(i,j)$, on peut, de même que pour le modèle de mélange, restreindre à $\mathrm{e}^{-d_{\phi(\mathbf{x},\boldsymbol{\mu})}}$. Ceci simplifie les calculs et justifie une fois de plus l'utilité de la relation famille exponentielle / divergence de Bregman.

6. Modèles semi-markoviens cachés

L'utilisation des HMM est bien plus adaptée dans le cas de données dépendantes, comme c'est le cas pour des descripteurs de signaux audio, qui impliquent forcément une dépendance temporelle. On associera à chaque état du modèle markovien une classe, et l'on cherchera lors du partitionnement à segmenter le signal en portions contigües et homogènes.

Cependant l'hypothèse markovienne implique nécessairement que la durée passée dans un état (c'est-à-dire le nombre d'observations adjacentes dont la variable latente associée est dans le même état) suit une loi de probabilité implicite. En effet la probabilité de rester pendant une durée d dans l'état i est $\mathrm{p}(h_{t+1} = i, \ldots, h_{t+d} = i, h_{t+d+1} \neq i; \boldsymbol{\theta}) = a_{i,i}^{d-1}(1 - a_{i,i})$. C'est en fait la densité d'une loi géométrique $\mathcal{G}$ de paramètre $a_{i,i}$. La figure I.4 donne des exemples de densités de probabilité de lois géométriques.

Si dans certains cas de figure modéliser la durée des états par cette distribution peut s'avérer intéressant, cela devient plus problématique lorsque l'on cherche à définir une certaine durée comme ayant la plus forte probabilité. Les signaux audio étant par nature structurés, on pourrait notamment

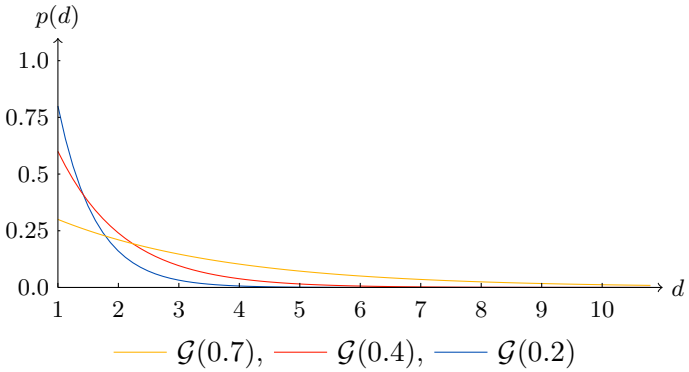


FIGURE I.4 – Densité de probabilité de la loi Géométrique

La durée la plus probable est toujours 1. Plus $a_{i,i}$ est faible, plus un tirage donnera des durées faibles, plus il est élevé plus les durées sont équiprobables.

associer les éléments structurant aux états en modélisant la durée de ces derniers de manière similaire à la durée de ces éléments.

L'idée est alors de conserver la modélisation markovienne pour les transitions entre états différents, et de modéliser la durée d'un état par une distribution appropriée. Ce modèle est dit semi-markovien et on parle alors de HSMM (*Hidden Semi Markov Models*) (GUÉDON 2003 ; MURPHY 2002 ; YU 2010).

1. Modèle général

Un modèle semi-markovien caché est une extension des HMM pour lequel la durée de séjour dans un état est une variable aléatoire d à valeurs dans $\mathbb{N}*$, qui est explicitement définie. On considère alors qu'on peut émettre non pas une observation mais une séquence d'observation dont le début et la fin déterminent d. À la fin d'une séquence le changement d'état s'effectue toujours selon une loi markovienne.

Le terme HSMM peut dénoter de nombreux types de modèles différents (voir YU (2010)), nous décrirons ici le modèle « HMM à durée explicite », pour lequel les transitions entre états ainsi que les émissions d'observations sont indépendantes des durées, et la durée d'un état est indépendante de l'état précédent. Pour plus de commodité, nous utiliserons dorénavant la notation de YU (2010) :

— $h_{[t_1:t_2]} = i \Leftrightarrow h_{t_1:t_2} = i$ et la séquence commence en t_1 et termine en t_2.
— $h_{[t_1:t_2} = i \Leftrightarrow h_{t_1:t_2} = i$ et la séquence commence en t_2, on ne sait pas quand elle termine.
— $h_{t_1:t_2]} = i \Leftrightarrow h_{t_1:t_2} = i$ et la séquence termine en t_2, on ne sait pas quand elle commence.

On définit alors pour chaque état i une probabilité de durée de densité $\mathrm{p}_i(d)$ de telle sorte que

$$\mathrm{p}(h_{t+2:t+d]} = i | h_{[t+1} = i; \boldsymbol{\theta}) = \mathrm{p}_i(d) \tag{I.26}$$

Les autres paramètres étant définis de la même manière que pour les HMM, le modèle est alors défini par $\boldsymbol{\theta} = (a, \boldsymbol{\pi}, \{\boldsymbol{\mu}_k\}_{k=1}^K, \{\mathrm{p}_k\}_{k=1}^K)$.

2. Du choix des distributions de durée

Le choix de la distribution p_i joue un rôle central pour les HSMM. Une solution pourrait être de choisir une loi non paramétrique, cependant on préfèrera en général utiliser des distributions paramétrique afin de n'avoir que quelques paramètres à ajuster lors de la phase d'apprentissage. Le tableau I.3 montre les détails de lois discrètes pouvant être utilisées comme distributions de durée. La loi de poisson est de famille exponentielle comme nous l'avons vu dans le tableau I.2. La loi binomiale

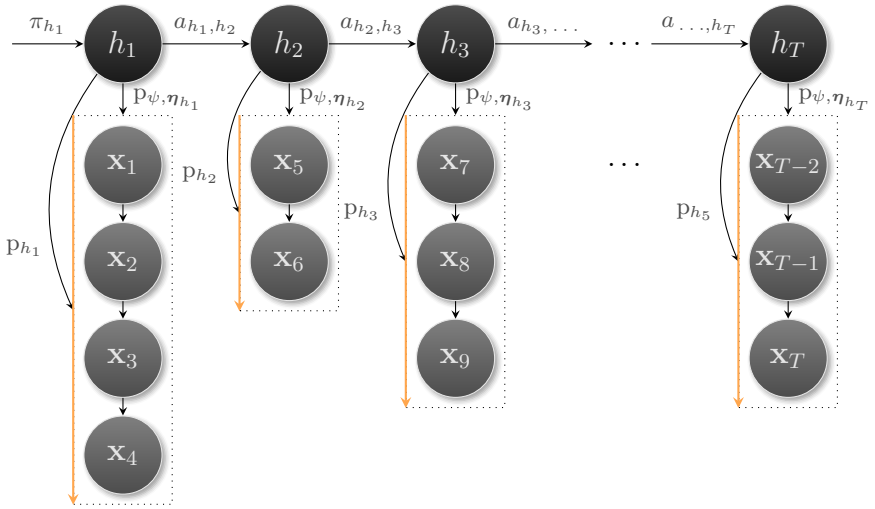


FIGURE I.5 – Représentation du modèle HSMM

Les séquences d'observations sont émises par un état. La durée d'une séquence émise par h_i, représentée par $\longrightarrow$, est déterminée par p_{h_i}. Les probabilités initiale π_i et de transition $a_{i,j}$ fonctionnent comme les HMM.

Nom	Paramètres	$\mathrm{p}(d)$	$\mathbb{E}[X]$	$\mathrm{Var}(X)$
Poisson	$\mathcal{P}(\lambda)$	$\mathrm{e}^{-\lambda}\frac{\lambda^d}{d!}$	λ	λ
Binomiale négative*	$\mathcal{N}(r,p)$	$\frac{\Gamma(r+d)}{d!\Gamma(r)}p^r(1-p)^d$	$\frac{r(1-p)}{p}$	$\frac{r(1-p)}{p^2}$

TABLE I.3 – Loi discrètes pouvant modéliser les durées

* Γ correspond à la fonction gamma $\Gamma(k) = \int_0^{+\infty} t^{k-1}\mathrm{e}^{-t}\,\mathrm{d}t$

négative peut aussi être considérée de famille exponentielle si on fixe le paramètre r. On verra que l'utilisation d'une famille exponentielle permet de généraliser l'apprentissage des paramètres de la même manière que pour les loi d'émission (Mitchell et al. 1993).

3. Inférence et estimation

Viterbi

L'algorithme de Viterbi peut facilement être étendu au nouveau modèle à durée explicite. Pour cela définissons les nouvelles valeurs semblables à celles des HMM (Yu 2010)

$$\delta_t(i) = \max_{h_{1:t}} \mathrm{p}(\mathbf{x}_{1:t}, h_{1:t-1}, h_{t]} = i; \boldsymbol{\theta}) \tag{I.27}$$

$$\delta_t^*(i) = \max_{h_{1:t+1}} \mathrm{p}(\mathbf{x}_{1:t}, h_{1:t}, h_{[t+1} = i; \boldsymbol{\theta}) \tag{I.28}$$

On obtient alors les expressions suivantes en procédant de la même manière.

$$\begin{aligned}
\delta_t(i) &= \max_{d\in\mathbb{N}^*} \max_{h_{1:t-d}} \mathrm{p}(\mathbf{x}_{1:t}, h_{1:t-d}, h_{[t-d+1:t]} = i; \boldsymbol{\theta}) = \max_{d\in\mathbb{N}*} \mathrm{p}_{(\psi,\boldsymbol{\eta}_i)}(\mathbf{x}_{t-d:t})\mathrm{p}_i(d)\delta_{t-d}^*(i) \\
\gamma_t(i) &= \arg\max_{d\in\mathbb{N}^*} \mathrm{p}_{(\psi,\boldsymbol{\eta}_i)}(\mathbf{x}_{t-d:t})\mathrm{p}_i(d)\delta_{t-d}^*(i) \\
\delta_t^*(j) &= \max_{i\in[\![1;K]\!]} \max_{h_{1:t-1}} \mathrm{p}(\mathbf{x}_{1:t}, h_{1:t-1}, h_{t]} = i, h_{t+1} = j; \boldsymbol{\theta}) = \max_{i\in[\![1;K]\!]} a_{i,j}\delta_t(i) \\
\gamma_t^*(j) &= \arg\max_{i\in[\![1;K]\!]} a_{i,j}\delta_t(i)
\end{aligned}$$

Avec $\delta_0^*(i) = \pi_i$. L'algorithme est alors mis en place de la même manière que pour les HMM, on calcule l'état final $h_T^* = \arg\max_i \delta_T(i)$ et grâce à des pointeurs $\gamma_t^*(i)$ donnant la longueur de la séquence finissant à t en i, et $\gamma_{t-d}^*(j)$ donnant l'état précédant cette séquence, on retrouve l'état initial. Notons tout de même que l'algorithme suppose que la dernière séquence finit exactement en T, ce qui peut poser problème dans le cas d'un HSMM avec un ou plusieurs états absorbants.

Forward-backward

L'algorithme *forward-backward* quant à lui s'étend aussi facilement aux HSMM en dédoublant les variables de la même manière que pour Viterbi, une avant changement d'état et une après changement d'état.

$$\alpha_t(i) = \mathrm{p}(h_{t]} = i, \mathbf{x}_{1:t}; \boldsymbol{\theta}) \tag{I.29}$$

$$\alpha_t^*(i) = \mathrm{p}(h_{[t+1} = i, \mathbf{x}_{1:t}; \boldsymbol{\theta}) \tag{I.30}$$

$$\beta_t(i) = \mathrm{p}(\mathbf{x}_{t+1:T}|h_{t]} = i; \boldsymbol{\theta}) \tag{I.31}$$

$$\beta_t^*(i) = \mathrm{p}(\mathbf{x}_{t+1:T}|h_{[t+1} = i; \boldsymbol{\theta}) \tag{I.32}$$

$$\tag{I.33}$$

On peut aussi, comme pour le modèle HMM, écrire ces valeurs de manière récursive, en décom-

posant les probabilités (L. Rabiner 1989).

$$\alpha_t(i) = \sum_{d\in\mathbb{N}^*} \mathrm{p}(h_{[t-d+1:t]} = i, \mathbf{x}_{1:t}; \boldsymbol{\theta}) = \sum_{d\in\mathbb{N}^*} \mathrm{p}_{(\psi,\boldsymbol{\eta}_i)}(\mathbf{x}_{t-d+1:t})\mathrm{p}_i(d)\alpha^*_{t-d}(i)$$

$$\alpha^*_t(j) = \sum_{i=1}^{K} \mathrm{p}(h_{t+1} = j, h_t = i, x_{1:t}; \boldsymbol{\theta}) = \sum_{i=1}^{K} a_{i,j}\alpha_t(i)$$

$$\beta_t(i) = \sum_{j=1} K\mathrm{p}(\mathbf{x}_{t+1:T}, h_{t+1} = j | h_{t]} = i; \boldsymbol{\theta}) = \sum_{j=1}^{K} \beta^*_t(t)a_{i,j}$$

$$\beta^*_t(j) = \sum_{d\in\mathbb{N}^*} \mathrm{p}(\mathbf{x}_{t+1:T}, h_{t+2:t+d]} = i | h_{[t+1} = i; \boldsymbol{\theta}) = \sum_{(d\in\mathbb{N}^*} \mathrm{p}_{\psi,\boldsymbol{\eta}_i)}(\mathbf{x}_{t+1:t+d})\beta_{t+d}(j)\mathrm{p}_j(d)$$

Avec pour valeurs initiales $\alpha^*_0(i) = \mathrm{p}(h_1 = i; \boldsymbol{\theta}) = \pi_i$ et $\beta_T(i) = 1$. En pratique on devra fixer la durée maximale d'un état à D pour faire les sommes sur d. On obtient alors une complexité temporelle $O(T(KD + K^2))$ et une complexité spatiale de $O(TK)$

Note Le modèle proposé ici considère que lorsqu'une séquence émise par un état est finie, l'état suivant peut être le même, ceci peut être utile dans certaines situations. On peut interdire une transition vers le même état, il suffit alors d'imposer $i \neq j$ dans le calcul des variables ci-dessus, ou de forcer $a_{i,i} = 0$ pour les états en question.

Inférences

Avec les valeurs données par l'algorithme *forward-backward*, on peut alors calculer les valeurs suivantes

$$\begin{aligned}
\mathrm{p}(h_{t]} = i, \mathcal{X}) &= \mathrm{p}(\mathbf{x}_{t+1:T}|h_{t]} = i)\mathrm{p}(\mathbf{x}_{1:t}, h_{t]} = i) = \beta_t(i)\alpha_t(i) \\
\mathrm{p}(h_{[t+1} = i, \mathcal{X}) &= \mathrm{p}(\mathbf{x}_{t+1:T}|h_{[t+1} = i)\mathrm{p}(\mathbf{x}_{1:t}, h_{[t+1} = i) = \beta^*_t(i)\alpha^*_t(i) \\
\mathrm{p}(h_{t]} = i, h_{[t+1} = j, \mathcal{X}) &= \mathrm{p}(\mathbf{x}_{t+1:T}|h_{[t+1} = j)\mathrm{p}(h_{[t+1} = j|h_{t]} = i)\mathrm{p}(\mathbf{x}_{1:t}, h_{t]} = i) \\
&= \beta^*_t(j)a_{i,j}\alpha_t(i) \\
\mathrm{p}(h_{[t+1:t+d]} = i, \mathcal{X}) &= \mathrm{p}(\mathbf{x}_{t+d+1:T}|h_{t+d]} = i)\mathrm{p}(\mathbf{x}_{t+1:t+d}|h_{[t+1:t+d]} = i)\mathrm{p}(h_{t+2:t+d]} = i|h_{[t+1} = i) \\
&\qquad \mathrm{p}(\mathbf{x}_{1:t}, h_{[t+1} = i) \\
&= \beta_{t+d}(i)\mathrm{p}_{(\psi,\boldsymbol{\eta}_i)}(\mathbf{x}_{t+1:t+d})\mathrm{p}_i(d)\alpha^*_t(i)
\end{aligned}$$

Puisqu'on estime que la première séquence démarre en $t = 1$, et donc que $\sum_{i=1}^{K} \mathrm{p}(h_{[1} = i) = 1$, on peut calculer $\mathrm{p}(\mathcal{X}) = \sum_{i=1}^{K} \mathrm{p}(h_{[1}, \mathcal{X}) = \sum_{i=1}^{K} \beta^*_0(i)\alpha^*_0(i)$. En divisant les statistiques précédentes par cette valeur, on obtient alors les variables suivantes, qui nous servirons pour la mise à jour du modèle.

$$\begin{aligned}
\delta_t(i) &= \mathrm{p}(h_{t]} = i|\mathcal{X}, \boldsymbol{\theta}) = \beta_t(i)\alpha_t(i)/\mathrm{p}(\mathcal{X}) \\
\delta^*_{t+1}(i) &= \mathrm{p}(h_{[t+1} = i|\mathcal{X}, \boldsymbol{\theta}) = \beta^*_t(i)\alpha^*_t(i)/\mathrm{p}(\mathcal{X}) \\
\epsilon_t(i,j) &= \mathrm{p}(h_{t]} = i, h_{[t+1} = j|\mathcal{X}) = \beta^*_t(j)a_{i,j}\alpha_t(i)/\mathrm{p}(\mathcal{X}) \\
l_{t,d}(i) &= \mathrm{p}(h_{[t+1:t+d]} = i|\mathcal{X}) = \beta_{t+d}(i)\mathrm{p}_{(\psi,\boldsymbol{\eta}_i)}(\mathbf{x}_{t+1:t+d})\mathrm{p}_i(d)\alpha^*_t(i)/\mathrm{p}(\mathcal{X})
\end{aligned}$$

De plus en remarquant que $\mathrm{p}(h_{t+1} = i) = \mathrm{p}(h_t = i) - \mathrm{p}(h_{t]} = i) + \mathrm{p}(h_{t]} = i, h_{[t+1} = i) + \mathrm{p}(h_{[t+1} = i)$ on peut déterminer

$$\tau_t(i) = \mathrm{p}(h_t = i|\mathcal{X}) = \sum_{t'<t} \delta^*_{t'}(i) + \epsilon_{t'}(i,i) - \delta_{t'}(i)$$

Le terme en ϵ n'est utile que si on considère les transitions vers le même état à la fin d'une séquence possibles.

4. Espérance-Maximisation

Afin de pouvoir ré-estimer les paramètres, calculons la variable $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(\tau)}) = \mathbb{E}\Big[\ln(\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta}))|\mathcal{X}; \boldsymbol{\theta}^{(\tau)}\Big]$ définie par l'algorithme EM.

$$\sum_{t=1}^{T}\sum_{i=1}^{K}\tau_t^{(\tau)}(i)\ln(\mathrm{p}_{(\psi,\boldsymbol{\eta}_i)}(\mathbf{x}_i)) + \sum_{t=1}^{T}\sum_{d=1}^{T-t}\sum_{i=1}^{K} l_{t,d}^{(\tau)}(i)\ln(\mathrm{p}_i(d)) + \sum_{t=1}^{T}\sum_{i=1}^{K}\sum_{j=1}^{K}\epsilon_t^{(\tau)}(i,j)\ln(a_{i,j}) + \sum_{i=1}^{K}\tau_1^{(\tau)}(i)\ln(\pi_i)$$

On en déduit que la partie maximisation est identique aux HMM pour $\boldsymbol{\mu}_i$, π_i et $a_{i,j}$

$$\begin{aligned}
\pi_i^{(\tau+1)} &= \tau_1^{(\tau)}(i)\\
a_{i,j}^{(\tau+1)} &= \frac{\sum_{t=1}^{T}\epsilon_t^{(\tau)}(i,j)}{\sum_{t=1}^{T}\delta_t^{(\tau)}(i)}\\
\boldsymbol{\mu}_i^{(\tau+1)} &= \frac{\sum_{t=1}^{T}\tau_t^{(\tau)}(i)\mathbf{x}_i}{\sum_{t=1}^{T}\tau_t^{(\tau)}(i)}
\end{aligned}$$

Le deuxième terme de la somme dans $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(\tau)})$ va nous permettre de maximiser la probabilité de durée p_i. Dans le cas d'une distribution non paramétrique, où la probabilité de durée est définie par le vecteur $\mathbf{p}_{i,D} = [\mathrm{p}_i(1), \ldots, \mathrm{p}_i(D)]$, puisqu'on a $\sum_d \mathrm{p}_i(d) = 1$, on peut maximiser ce vecteur comme on a maximisé $\boldsymbol{\pi}$ pour le modèle de mélange (I.15), où le vecteur $(\mathbf{A})_i$ pour les HMM (I.25). En remarquant que $\sum_d l_{t,d}^{(\tau)}(i) = \delta_t^{*(\tau)}(i)$. On trouve

$$\mathrm{p}_i^{(\tau+1)}(d) = \frac{\sum_{t=1}^{T} l_t^{(\tau)}(i,d)}{\sum_{t=1}^{T}\delta_t^{*(\tau)}(i)} \tag{I.34}$$

Dans le cas où la distribution de durée est une famille exponentielle (MITCHELL et al. 1993), on peut mettre à jour les paramètres en écrivant sa densité en fonction d'une divergence de Bregman, de la même manière qu'on l'a fait pour mettre à jour les paramètres $\boldsymbol{\mu}_i$. Si $\mathrm{p}_i = \mathrm{p}_{\psi_D,\boldsymbol{\eta}_{D,i}}$ où on a la dualité $(\psi_D, \boldsymbol{\eta}_{D,i}) \leftrightarrow (\phi_D, \boldsymbol{\mu}_{D,i})$ alors en suivant le raisonnement amenant à (I.14), on a

$$\begin{aligned}
\boldsymbol{\mu}_{D,i}^{(\tau+1)} &= \arg\min_{\boldsymbol{\mu}_{D,i}} \sum_{t=1}^{T}\sum_{\mathbf{d}} l_{t,d}^{(\tau)}(i) d_{\phi_D}(\mathbf{d}, \boldsymbol{\mu}_{D,i})\\
&= \frac{\sum_{\mathbf{d}}\sum_{t=1}^{T} l_{t,d}^{(\tau)}(i)\mathbf{d}}{\sum_{t=1}^{T}\delta_t^{*(\tau)}(i)}
\end{aligned} \tag{I.35}$$

Cela revient en fait à ré-estimer le paramètre comme l'espérance de d étant données les probabilités estimées par (I.34) : $\boldsymbol{\mu}_{D,i}^{(\tau)} = \mathbb{E}_{\mathrm{p}_i^{(\tau+1)}}[d]$

Chapitre II

Partitionnement incrémental

Les algorithmes de partitionnement présentés précédemment supposent que l'on connaisse l'intégralité des données pour apprendre le modèle. Ce type de partitionnement est qualifié de *batch clustering*. Ce prérequis peut être problématique pour certaines applications où l'on désire pouvoir apprendre les paramètres et partitionner les données au fur et à mesure que celles-ci sont reçues – par exemple dans le cas de l'analyse et la détection d'événements pour des scènes acoustiques. Traiter le partitionnement présente de plus l'avantage d'être moins coûteux en mémoire, puisqu'il n'est pas nécessaire de conserver l'ensemble des observations, et permet une éventuelle version de l'algorithme en temps réel, dans le cas où celui-ci est suffisamment rapide.

Ces algorithmes sont catégorisés d'algorithmes en ligne (*online*). Nous proposons ici d'étendre l'algorithme EM exposé en section I-4.2 pour les trois modèles présentés : le modèle de mélange, le modèle HMM, et le modèle HSMM. Deux types d'algorithmes sont présentés pour chaque modèle, une version que l'on qualifiera simplement de *en ligne*, et une version qualifiée de *incrémentale*. Pour chaque version, nous présenterons l'algorithme adapté aux modèles de mélange, HMMet HSMM.

1. Algorithme en ligne, modèle de mélange

La version de l'algorithme de l'algorithme EM que nous décrivons ici en ligne est basé sur des outils d'optimisation, permettant une écriture de la partie maximisation utilisant une approximation par la méthode de Newton (Lange 1995). Selon Cappé et Moulines (2009), la première version en ligne de l'algorithme d'estimation des paramètres est proposée par Titterington (1984). Cependant, cet algorithme, basé sur une approximation stochastique inspirée de la méthode précédente, nécessite des calculs d'inversion de matrices Hessiennes, et offre peu de ressemblance avec l'algorithme original.

L'algorithme en ligne proposé ici effectue l'approximation stochastique lors de l'étape d'espérance plutôt que lors de l'étape de Maximisation (Cappé et Moulines 2009), et garde cette dernière étape inchangée. Le calcul de Q devient alors une récurrence déterminée par

$$\hat{Q}^{(i+1)}(\boldsymbol{\theta}) = \hat{Q}^{(i)}(\boldsymbol{\theta}) + \gamma_{i+1}\left(\mathbb{E}\Big[\ln(\mathrm{p}(\mathbf{x}_{i+1}, h_{i+1}; \boldsymbol{\theta}))|\mathcal{X}; \boldsymbol{\theta}^{(i)}\Big] - \hat{Q}^{(i)}(\boldsymbol{\theta})\right) \tag{II.1}$$

Notons que les itérations et les observations sont maintenant indexées toutes deux par i. La fonction $\gamma : \mathbb{N}* \mapsto [0, 1]$ dénote le pas et dépend de l'itération. En utilisant un modèle de mélange de famille exponentielle, on va pouvoir expliciter cette récursion et en déduire l'étape de maximisation.

1. Réécriture du modèle

Comme dans le chapitre précédent, nous nous restreignons aux modèles de mélange de famille exponentielles, à la lumière des propriétés énoncées.Une autre propriété intéressante que possède ce type de modèle de mélange est que la loi jointe d'une observation et une variable latente peut elle-même s'écrire sous la forme d'une famille exponentielle. En effet soit une famille exponentielle déterminée par ψ et un modèle de mélange défini par $\boldsymbol{\theta} = (\boldsymbol{\pi}, \{\boldsymbol{\mu}_k\}_{k=1}^K$, telle la section I-4.1, on sait

déjà que pour une observation $\mathbf{x}$ et une variable cachée $h = k$, on a $\mathrm{p}(\mathbf{x}, h = k; \boldsymbol{\theta}) = \pi_k \exp(\langle \mathbf{x}, \boldsymbol{\eta}_k \rangle - \psi(\boldsymbol{\eta}_k))a(\mathbf{x})$. On obtient pour tout $h \in [\![1; K]\!]$

$$\mathrm{p}(\mathbf{x}, h; \boldsymbol{\theta}) = \exp\left(\sum_{k=1}^{K} \delta_{k,h}[\ln(\pi_k) + \langle \mathbf{x}, \boldsymbol{\eta}_k \rangle - \psi(\boldsymbol{\eta}_k)]\right) a(\mathbf{x})$$

où $\delta_{i,j}$ est le symbole de Kronecker [1]. En notant pour $k \in [\![1; K]\!]$ les statistiques exhaustives $S_k(\mathbf{x}, h) = [\delta_{k,h}, \delta_{k,h}\mathbf{x}]$ et $\phi_k(\boldsymbol{\theta}) = [\ln(\pi_k) - \psi(\boldsymbol{\eta}_k), \boldsymbol{\eta}_k]$ puis $S(\mathbf{x}, h) = [S_1(\mathbf{x}, h), \ldots, S_K(\mathbf{x}, h)]$ et $\phi(\boldsymbol{\theta}) = [\phi_1(\boldsymbol{\theta}), \ldots, \phi_K(\boldsymbol{\theta})]$, on a alors

$$\mathrm{p}(\mathbf{x}, h; \boldsymbol{\theta}) = \exp(\langle S(\mathbf{x}, h), \phi(\boldsymbol{\theta}) \rangle) a(\mathbf{x})$$

L'étape de maximisation de l'algorithme (section I-4.2) peut alors être exprimée uniquement en fonction des statistiques exhaustives, en utilisant la linéarité de l'espérance.

$$\arg\max_{\boldsymbol{\theta}} \mathbb{E}\Big[\ln(\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta}))|\mathcal{X}; \boldsymbol{\theta}^{(\prime)}\Big] = \arg\max_{\boldsymbol{\theta}} \left\langle \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}\Big[S(\mathbf{x}_i, h_i)|\mathcal{X}; \boldsymbol{\theta}^{(\prime)}\Big], \phi(\boldsymbol{\theta}) \right\rangle \tag{II.2}$$

On appelle [2] alors $\hat{s}^{(\tau)} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\mathbf{h}|\mathcal{X};\boldsymbol{\theta}^\tau}[S(\mathbf{x}_i, h_i)]$. Le paramètre maximal $\hat{\boldsymbol{\theta}}^{(\tau)}$ est alors déterminé par $\bar{\boldsymbol{\theta}}(\hat{s}^{(\tau)})$ où $\bar{\boldsymbol{\theta}}(\mathbf{s}) = \arg\max_{\boldsymbol{\theta}} \langle \mathbf{s}, \phi(\boldsymbol{\theta}) \rangle$. Si on note $\hat{s}_k^{(\tau)}$ le sous vecteur de $\hat{s}^{(\tau)}$ à l'indice k, c'est à dire déterminé par S_k, on obtient

$$\hat{s}_k^{(\tau)} = \frac{1}{n} \sum_{i=1}^{n} \sum_{k'}^{K} \tau_i^{(\tau)}(k') S_k(\mathbf{x}_i, k') = \frac{1}{n} \sum_{i=1}^{n} \begin{bmatrix} \tau_i^{(\tau)}(k) \\ \tau_i^{(\tau)}(k)\mathbf{x}_i \end{bmatrix}$$

où $\tau_i^{(\tau)}(k) = \mathrm{p}(h_i = k|\mathbf{x}_i; \boldsymbol{\theta})$. On constate alors d'après (I.15) et (I.14) qu'on obtient les mises à jour $\pi_k^{(\tau+1)} = \hat{s}_{k,1}^{(\tau)}$ et $\boldsymbol{\mu}_k^{(\tau+1)} = \hat{s}_{k,2}^{(\tau)} / \hat{s}_{k,1}^{(\tau)}$.

La version en ligne de l'algorithme EM va alors consister à calculer cette statistique $\hat{s}^{(\tau)}$ de manière itérative au fur et à mesure que les nouvelles observations sont connues, afin de calculer les mises à jour des paramètres.

2. Algorithme

L'équation (II.1) peut donc être changée pour un mélange de famille exponentielle. Avec ce qu'on a décrit précédemment, on peut se contenter d'approximer la statistique $\hat{s}^{(i)}$ connaissant les observations $\mathbf{x}_{1:i}$ de la même manière. L'algorithme à l'itération i devient alors.

Espérance **Calcul de** $\tau_{i+1}^{(i)}(k)$ (équation (I.16))

$$\hat{s}^{(i+1)} \leftarrow (1 - \gamma_{i+1})\hat{s}^{(i)} + \gamma_{i+1} \sum_{k=1} \tau_{i+1}^{(i)}(k) S(x_{i+1}, k)$$

Maximisation

$$\pi_k^{(i+1)} \leftarrow \hat{s}_{k,1}^{(i+1)}$$

$$\boldsymbol{\mu}_k^{(i+1)} \leftarrow \hat{s}_{k,2}^{(i+1)} / \hat{s}_{k,1}^{(i+1)}$$

1. $\delta_{i,j} = 1$ si $i = j$, $\delta_{i,j} = 0$ sinon

2. on normalise l'expression par n pour anticiper le cas en ligne, où le nombre d'observations n va augmenter à chaque itération

2. HMM en ligne

On peut étendre ce procédé aux HMM (Cappé 2011). Soit un HMM défini par $\boldsymbol{\theta}$ tel que décrit en section I-5.1, puisque la loi jointe $\mathrm{p}(\mathcal{X}, \mathbf{h}; \boldsymbol{\theta}) = \mathrm{p}(\mathbf{x}_1, h_1; \boldsymbol{\theta}) \prod_{t>2} \mathrm{p}(\mathbf{x}_t, h_t | h_{t-1}; \boldsymbol{\theta})$. On va écrire la loi jointe *a priori* $\mathrm{p}(\mathbf{x}_t, h_t | h_{t-1}; \boldsymbol{\theta})$ sous forme d'une famille exponentielle comme on l'a fait pour le modèle de mélange.

$$\ln(\mathrm{p}(\mathbf{x}_t, h_t | h_{t-1}; \boldsymbol{\theta})) = \sum_{i=1}^{K}\sum_{j=1}^{K} \delta_{j,h_t}\delta_{i,h_{t-1}} \ln(a_{i,j}) \sum_{i=1}^{K} \delta_{j,h_t} \left[\langle \mathbf{x}_t, \boldsymbol{\eta}_i \rangle - \psi(\boldsymbol{\eta}_i)\right] + \ln(a(\mathbf{x})) \tag{II.3}$$

On peut alors définir la statistique S composée des vecteurs $S_i(\mathbf{x}_t, h_t) = [\delta_{i,h_t}, \delta_{i,h_t}\mathbf{x}]$ et $S_{i,j}(h_t|h_{t-1}) = [\delta_{j,h_t}\delta_{i,h_{t-1}}]$. La maximisation s'écrit donc de la même manière que (II.2). On calcule alors l'espérance de cette statistique

$$\hat{s}_i^{(\tau)} = \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} S_i(\mathbf{x}_t, h_t) | \mathcal{X}; \boldsymbol{\theta}^{(\tau)}\right] = \frac{1}{T}\sum_{t=1}^{T} \begin{bmatrix} \tau_t^{(\tau)}(i) \\ \tau_t^{(\tau)}(i)\mathbf{x}_t \end{bmatrix}$$

$$\hat{s}_{i,j}^{(\tau)} = \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} S_{i,j}(\mathbf{x}_t, h_t | h_{t-1}) | \mathcal{X}; \boldsymbol{\theta}^{(\tau)}\right] = \frac{1}{T}\sum_{t=1}^{T} \epsilon_t^{(\tau)}(i,j)$$

Où $\tau_t^{(\tau)}(i) = \mathrm{p}(h_t = i | \mathcal{X}; \boldsymbol{\theta}^{(\tau)})$ et $\epsilon_t^{(\tau)}(i,j) = \mathrm{p}(h_t = i, h_{t+1} = j | \mathcal{X}; \boldsymbol{\theta}^{(\tau)})$. D'après ce qu'on a vu sur la maximisation des paramètres HMM (section I-5.3), on a alors

$$\boldsymbol{\mu}_i^{(\tau+1)} = \hat{s}_{i,2}^{(\tau)} / \hat{s}_{i,1}^{(\tau)}$$
$$a_{i,j}^{(\tau+1)} = \hat{s}_{i,j}^{(\tau)} / \hat{s}_{i,1}^{(\tau)}$$

1. Calcul récursif des statistiques

Afin de pouvoir apprendre les paramètres en ligne, plutôt que d'utiliser l'algorithme *forward-backward*, Cappé (2011) présente une méthode pour calculer la probabilité dite de filtrage $\mathrm{p}(h_T|\mathcal{X})$ ainsi que la statistique $\hat{s}^{(\tau)}$ de manière incrémentale. On définit en effet

$$\phi_t(i) = \mathrm{p}(h_t = i | \mathbf{x}_{1:t}; \boldsymbol{\theta}) \tag{II.4}$$

$$\rho_t(i) = \frac{1}{t}\mathbb{E}\left[\sum_{l=1}^{t} S(\mathbf{x}_l, h_l | h_{l-1}) | \mathbf{x}_{1:t}, h_t = i; \boldsymbol{\theta}\right] \tag{II.5}$$

La somme $\sum_{i=1}^{K} \phi_n(i)\rho_n(i)$ donne la statistique $\hat{s}$ déterminée par les n premières observations.

On propose alors une écriture récursive de ces valeurs

Proposition II.1 *Pour les variables définies précédemment ((II.4) et (II.5)), on a*

$$\phi_1(i) = \frac{\pi_i \mathrm{e}^{-d_\phi(\mathbf{x}_1, \boldsymbol{\mu}_i)}}{\sum_{j=1}^{K} \pi_j \mathrm{e}^{-d_\phi(\mathbf{x}_1, \boldsymbol{\mu}_j)}}$$
$$\rho_1(i) = \mathbf{0}$$

et les formules de récurrence

$$\phi_{t+1}(j) = \frac{\sum_{i=1}^{K} \mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_{t+1}) a_{i,j}\phi_t(i)}{\sum_{i=1}^{K}\sum_{k=1}^{K} \mathrm{p}_{(\psi,\boldsymbol{\eta}_k)}(\mathbf{x}_{t+1}) a_{i,k}\phi_t(i)} \tag{II.6}$$

$$\rho_{t+1}(j) = \sum_{i=1}^{K} \left[\frac{1}{t+1} S(\mathbf{x}_{t+1}, h_{t+1} = j | h_t = i) + \left(1 - \frac{1}{t+1}\right)\rho_t(i)\right] r_t(i|j) \tag{II.7}$$

où

$$r_t(i|j) = \mathrm{p}(h_t = i|h_{t+1} = j, \mathbf{x}_{1:t}; \boldsymbol{\theta}) = \frac{a_{i,j}\phi_t(i)}{\sum_{k=1}^{K} a_{k,j}\phi_t(k)} \tag{II.8}$$

Preuve :

(II.6)
$$\begin{aligned}\mathrm{p}(h_{t+1} = j, \mathbf{x}_{t+1}|\mathbf{x}_{1:t}) &= \sum_i \mathrm{p}(h_{t+1} = j, h_t = i, \mathbf{x}_{t+1}|\mathbf{x}_{1:t}) \\ &= \sum_i \mathrm{p}(\mathbf{x}_{t+1}|h_{t+1} = j)\mathrm{p}(h_{t+1} = j|h_t = i)\mathrm{p}(h_t = i|\mathbf{x}_{1:t})\mathrm{p}(\mathbf{x}_{1:t}) \\ &= \sum_i \mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_{t+1})a_{i,j}\phi_t(i)\mathrm{p}(\mathbf{x}_{1:t})\end{aligned}$$

(II.7)
$$\begin{aligned}&\frac{1}{t+1}\mathbb{E}\left[\sum_{l=1}^{t+1} S(\mathbf{x}_l, h_l|h_{l-1})|\mathbf{x}_{1:t+1}, h_{t+1} = j; \boldsymbol{\theta}\right] \\ =&\frac{1}{t+1}\sum_i \mathbb{E}\left[\sum_{l=1}^{t+1} S(\mathbf{x}_l, h_l|h_{l-1})|\mathbf{x}_{1:t+1}, h_{t+1} = j, h_t = i; \boldsymbol{\theta}\right] \mathrm{p}(h_t = i|h_{t+1} = j, \mathbf{x}_{1:t}) \\ =&\frac{1}{t+1}\sum_i \left[S(\mathbf{x}_{t+1}, h_{t+1} = j|h_t = i) + \mathbb{E}\left[\sum_{l=1}^{t} S(\mathbf{x}_l, h_l|h_{l-1})|\mathbf{x}_{1:t+1}, h_t = i; \boldsymbol{\theta}\right]\right] r_t(i|j) \\ =&\sum_i \left[\frac{1}{t+1}S(\mathbf{x}_{t+1}, h_{t+1} = j|h_t = i) + (1 - \frac{1}{t+1})\rho_t(i)\right] r_t(i|j)\end{aligned}$$

(II.8)
$$\begin{aligned}\mathrm{p}(h_t = i, h_{t+1} = j|\mathbf{x}_{1:t}) =&\mathrm{p}(h_{t+1} = j|h_t = i)\mathrm{p}(h_t = i|\mathbf{x}_1^t) \\ =&a_{i,j}\phi_t(i)\end{aligned}$$

□

Ceci nous permet de calculer la statistique $\hat{s}$ pour chaque nouvelle observation $\mathbf{x}_t$ s'ajoutant au jeu de données. On peut alors en déduire une version en ligne de l'algorithme EM pour les HMMs.

2. Algorithme EM en ligne

On s'inspire de l'algorithme EM en ligne pour le modèle de mélange proposé par Cappé et Moulines (2009), et décrit en section II-1, ainsi que les formules de récursion (II.6) et (II.7). Il suffit alors de calculer à chaque itération t les nouvelles valeurs ϕ_{t+1} et ρ_{t+1} à partir des paramètres connus, et de remplacer le pas d'incrémentation $\frac{1}{t+1}$ par le pas d'incrémentation d'approximation stochastique γ_{t+1}. On peut alors en déduire $\hat{s}^{(t+1)}$ et donc les paramètres maximisant l'espérance. L'algorithme est présenté par l' algorithme 3. On peut estimer à chaque itération t la classe la plus probable pour $\mathbf{x}_t$ avec $\arg\max_j \phi_t(j)$

On détermine un temps minimal t_{min} avant que l'on applique l'apprentissage des paramètres. Son rôle est de s'assurer que la maximisation se déroule correctement sur le plan du calcul numérique. En effet pour un petit nombre d'observations l'estimation peut être dégénérée, et la fonction de de maximisation est alors incorrecte.

3. L'étape **Normalise** de l'algorithme indique que l'on divise la valeur par la somme de ses composantes. Par exemple **Normalise** $\phi_{t+1} \Leftrightarrow \phi_{t+1}(i) \leftarrow \phi_{t+1}(i)/\sum_j \phi_{t+1}(j)$

Algorithme 3 Espérance-Maximisation en ligne pour HMM [3]

Procédure HMM EN LIGNE($\mathbf{x}_t$ à l'instant t, paramètre $\boldsymbol{\theta}^{(1)}$ initial, $\gamma : \mathbb{N}^* \mapsto [0, 1[$, t_{min})

$\phi_1(i) \leftarrow \pi_i \mathrm{e}^{-d_\phi(\mathbf{x}_1, \boldsymbol{\mu}_i^{(1)})}$, **Normalise** ϕ_1

$\rho_1(i) \leftarrow \mathbf{0}$

Pour $t \leftarrow 1, \ldots, T-1$ **faire**

Espérance:

$$
\begin{aligned}
r_t(i|j) &\leftarrow a_{i,j}^{(t)} \phi_t(i), \quad \textbf{Normalise } r_t(\cdot|j) \\
\phi_{t+1}(j) &\leftarrow \textstyle\sum_i \mathrm{p}_{(\psi, \boldsymbol{\eta}_j^{(t)})}(\mathbf{x}_{t+1}) a_{i,j}^{(t)} \phi_t(i), \quad \textbf{Normalise } \phi_{t+1} \\
\rho_{t+1}(j) &\leftarrow \textstyle\sum_i \left[\gamma_{t+1} S(\mathbf{x}_{t+1}, h_{t+1} = j | h_t = i) + (1 - \gamma_{t+1})\, \rho_t(i)\right] r_t(i|j) \\
\hat{s}^{(t+1)} &\leftarrow \textstyle\sum_j \phi_{t+1}(j) \rho_{t+1}(j)
\end{aligned}
$$

Maximisation:

Si $t > t_{min}$ **then**

$$
\begin{aligned}
\boldsymbol{\mu}_i^{(t+1)} &\leftarrow \hat{s}_{i,2}^{(t+1)} / \hat{s}_{i,1}^{(t+1)} \\
a_{i,j}^{(t+1)} &\leftarrow \hat{s}_{i,j}^{(t+1)} / \hat{s}_{i,1}^{(t+1)}
\end{aligned}
$$

Fin Si

Fin Pour

Retourne $\boldsymbol{\theta}^{(T)}, \arg\max_j \phi_t(j)\ \forall t$

Fin Procédure

Calcul de ρ_t Le vecteur ρ_t possède le même nombre de composantes que la statistique $S(\mathbf{x}_t, h_t|h_{t-1})$. On peut alors expliciter la formule de mise à jour de chaque composante. En effet $S_k(\mathbf{x}_t, h_t) = \delta_{k,h_t}[1, \mathbf{x}_t]$ et $S_{k,k'}(h_t|h_{t-1}) = \delta_{k,h_{t-1}}\delta_{k',h_t}$. On en déduit que les composantes correspondantes de ρ_t ont pour mise à jour

$$
\begin{aligned}
\rho_{t+1}^k(j) &\leftarrow \gamma_{t+1}\delta_{j,k}[1, \mathbf{x}_{t+1}] + (1 - \gamma_{t+1}) \sum_i \rho_t^k(i) r(i|j) \\
\rho_{t+1}^{k,k'}(j) &\leftarrow \gamma_{t+1}\delta_{j,k'} r(k|j) + (1 - \gamma_{t+1}) \sum_i \rho_t^{k,k'} r(i|j)
\end{aligned}
$$

Complexité Déterminons la complexité de l'algorithme pour une itération, en notant K le nombre de classes et n le nombre de dimensions de la statistique $\mathbf{x}_t$. Le calcul de ϕ_t nécessite un vecteur de dimension K, donc une complexité spatiale en $O(K)$ et d'après la formule de récurrence, une complexité temporelle en $O(K^2)$. Le calcul de r_t nécessite lui un vecteur de dimension K^2 et a une complexité temporelle en $O(K^2)$. L'ensemble des composantes de ρ_t impliquent une complexité spatiale en $O(K^2(n+1) + K^3)$, et les formules de récurrence impliquent une complexité temporelle en $O(K^3(n+1) + K^4)$. Finalement on a donc pour une itération t

Complexité spatiale $O(K^2 n + K^3)$

Complexité temporelle $O(K^3 n + K^4)$

3. HSMM en ligne

Pour l'algorithme en ligne d'apprentissage du modèle HMM, on va suivre le même raisonnement. On définit avant tout une nouvelle variable aléatoire d_t, qui détermine à l'instant t depuis combien de temps la séquence émise a démarré. Formellement on a alors l'équivalence $h_i, d_t = d \Leftrightarrow h_{[t-d+1:t} = i$.

On utilise alors une statistique qui nous permettra de mettre à jour les paramètres. On définit alors $S(\mathbf{x}_t, h_t, d_t|h_{t-1}, d_{t-1})$. Cette statistique est composée des éléments suivants :

$$\begin{aligned} S_i(\mathbf{x}_t, h_t) &= [\delta_{i,h_t}, \delta_{i,h_t}\mathbf{x}] \\ S_{i,j}(h_t, d_t|h_{t-1}) &= \delta_{j,h_t}\delta_{i,h_{t-1}}\delta_{d_t,1} \\ S_{i,d}(d_t|h_{t-1}, d_{t-1}) &= \delta_{i,h_{t-1}}\delta_{d,d_{t-1}}\delta_{d_t,1} \end{aligned} \tag{II.9}$$

Le calcul $\hat{s}^{(\tau)} = \frac{1}{T}\mathbb{E}\Big[S(\mathbf{x}_t, h_t, d_t|h_{t-1}, d_{t-1})|\mathcal{X}; \boldsymbol{\theta}^{(\tau)}\Big]$ nous permet alors de maximiser les paramètres. On sait déjà que $\hat{s}^{(\tau)}_{i,2}/\hat{s}^{(\tau)}_{i,1}$ nous permet de calculer $\boldsymbol{\mu}_i^{(\tau+1)}$. Pour les autres paramètres, on calcule.

$$\begin{aligned} \hat{s}^{(\tau)}_{i,j} &= \frac{1}{T}\sum_t\sum_{i'}\sum_{j'}\epsilon_t^{(\tau)}(i',j')\delta_{i,i'}\delta_{j,j'} = \frac{1}{T}\sum_t \epsilon_t^{(\tau)}(i,j) \\ \hat{s}^{(\tau)}_{i,d} &= \frac{1}{T}\sum_t\sum_{i'}\sum_{d'} l_{t,d'}^{(\tau)}(i')\delta_{i,i'}\delta_{d,d'} = \frac{1}{T}\sum_t l_{t,d}^{(\tau)}(i) \end{aligned}$$

où $\epsilon_t^{(\tau)}(i,j) = \mathrm{p}(h_{t-1]} = i, h_{[t} = j|\mathbf{x}_{1:T}; \boldsymbol{\theta}^{(\tau)})$ et $l_{t,d}^{(\tau)}(i) = \mathrm{p}(h_{[t-d:t-1]}|\mathbf{x}_{1:T}; \boldsymbol{\theta}^{(\tau)})$ comme définies en section I-6.3. D'après la section I-6.4, on en déduit la maximisation des paramètres :

$$\begin{aligned} a_{i,j}^{(\tau+1)} &= \hat{s}^{(\tau)}_{i,j}/\sum_k \hat{s}^{(\tau)}_{i,k} \\ \mathrm{p}_i^{(\tau+1)}(d) &= \hat{s}^{(\tau)}_{i,d}/\sum_{d'} \hat{s}^{(\tau)}_{i,d'} \quad \text{si la distribution de durée est non paramétrique} \\ \boldsymbol{\mu}_{D,i}^{(\tau+1)} &= \sum_d \hat{s}^{(\tau)}_{i,d} d/\sum_d \hat{s}^{(\tau)}_{i,d} \quad \text{si paramétrique de paramètre } \boldsymbol{\mu}_{D,i} \end{aligned}$$

1. Calcul récursif

Comme précédemment on va définir deux variables, pouvant être définies récursivement, qui vont nous permettre de calculer $\hat{s}$. Contrairement au HMM, nous avons pour chaque temps t deux variables aléatoires latentes : h_t et d_t. Les deux valeurs récursives ϕ_t et ρ_t dépendent donc maintenant de deux variables, et deviennent donc

$$\begin{aligned} \phi_t(i,d) &= \mathrm{p}(h_t = i, d_t = d|\mathbf{x}_{1:t}; \boldsymbol{\theta}) \\ \rho_t(i,d) &= \frac{1}{t}\mathbb{E}\left[\sum_{l=1}^{t} S(\mathbf{x}_l, h_l, d_l|h_{l-1}, d_{l-1})|h_t = i, d_t = d; \mathbf{x}_{1:t}; \boldsymbol{\theta}\right] \end{aligned}$$

De même que pour les HMM, on a alors l'égalité $\sum_i\sum_d \phi_t(i,d)\rho_t(i,d) = \hat{s}$ avec $\hat{s}$ l'espérance de la statistique déterminée par les t premières observations. La proposition II.2 détermine alors l'écriture récursive de ces deux variables.

Proposition II.2 *Soit ϕ_t et ρ_t définie ci-dessus, on définit $\bar{\phi}_t(i,d) = \mathrm{p}(h_t = i, d_t = d, \mathbf{x}_{1:t}; \boldsymbol{\theta})$ et $D_i(d) = \mathrm{p}(h_{[it-d+1:t} = i|h_{[ti-d+1} = i; \boldsymbol{\theta}) = \sum_{d'>d}\mathrm{p}_i(d')$. Alors*

$$\begin{aligned} \phi_1(i,1) &= \frac{\pi_i \mathrm{e}^{-d_\phi(\mathbf{x}_1, \boldsymbol{\mu}_i^{(1)})}}{\sum_j \pi_j \mathrm{e}^{-d_\phi(\mathbf{x}_1, \boldsymbol{\mu}_j^{(1)})}} \\ \phi_1(i,d) &= 0 \quad \textit{pour } d > 1 \\ \rho_1(i,d) &= \mathbf{0} \end{aligned}$$

et $\forall t \in [2;T]$

$$\bar{\phi}_{t+1}(j,1) = \sum_i \sum_d \mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_{t+1}) a_{i,j} \left(1 - \frac{D_i(d+1)}{D_i(d)}\right) \bar{\phi}_t(i,d) \tag{II.10}$$

$$\bar{\phi}_{t+1}(j,d) = \mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_{t+1}) \frac{D_j(d)}{D_j(d-1)} \bar{\phi}_t(j,d-1) \quad \text{si } d > 1 \tag{II.11}$$

$$\phi_{t+1}(j,d) = \frac{\bar{\phi}_{t+1}(j,d)}{\sum_{j'} \sum_{d'} \bar{\phi}_{t+1}(j',d')} \tag{II.12}$$

$$\rho_{t+1}(j,1) = \sum_i \sum_d \left(\frac{1}{t+1} S(\mathbf{x}_{t+1}, j, 1 | i, d) + \left(1 - \frac{1}{t+1}\right) \rho_t(i,d)\right) r_{t+1}(i,d|j) \tag{II.13}$$

$$\rho_{t+1}(j,d) = \frac{1}{t+1} S(\mathbf{x}_{t+1}, j, d | j, d-1) + \left(1 - \frac{1}{t+1}\right) \rho_t(j,d-1) \quad \text{si } d > 1 \tag{II.14}$$

avec

$$\begin{aligned} r_{t+1}(i,d|j) &= \mathrm{p}(h_t = i, d_t = d | h_{t+1} = j, d_{t+1} = 1, \mathbf{x}_{1:t}) \\ &= \frac{a_{i,j} \left(1 - \frac{D_i(d+1)}{D_i(d)}\right) \phi_t(i,d)}{\sum_k \sum_{d'} a_{k,j} \left(1 - \frac{D_k(d'+1)}{D_k(d')}\right) \phi_t(k,d')} \end{aligned} \tag{II.15}$$

Preuve : Résultat préliminaire : on montre d'abord que pour $d > 1$,

$$\begin{aligned} \mathrm{p}(h_{[t-d+1:t} = i | h_{[t-d+1:t-1} = i) &= \frac{\mathrm{p}(h_{[t-d+1:t} = i | h_{[t-d+1} = i)}{\mathrm{p}(h_{[t-d+1:t-1} = i | h_{[t-d+1} = i)} \\ &= \frac{D_i(d)}{D_i(d-1)} \end{aligned}$$

La valeur $D_i(d)$ désignant la probabilité qu'une séquence ait une durée supérieure ou égale à d. Le rapport entre ces deux valeurs successives désigne donc la probabilité qu'il n'y ait pas de transition entre $t-1$ et t sachant que la séquence est pour l'instant de longueur $d-1$.

De plus, la probabilité qu'une séquence se termine en t sachant qu'elle est déjà de longueur d vaut alors

$$\begin{aligned} \mathrm{p}(h_{[t-d+1:t]} = i | h_{[t-d+1:t} = i) &= 1 - \mathrm{p}(h_{[t-d+1:t+1} = i | h_{[t-d+1:t} = i) \\ &= \left(1 - \frac{D_i(d+1)}{D_i(d)}\right) \end{aligned}$$

On en déduit alors les récursions.

(II.10)

$$\begin{aligned} &\mathrm{p}(h_{t+1} = j, d_t = 1, \mathbf{x}_{1:t+1}) = \mathrm{p}(h_{[t+1} = j, \mathbf{x}_{1:t+1}) \\ =&\sum_i \sum_d \mathrm{p}(h_{[t+1} = j, h_{[t-d+1:t]} = i, , \mathbf{x}_{1:t+1}) \\ =&\sum_i \sum_d \mathrm{p}(\mathbf{x}_{t+1} | h_{t+1} = j) \mathrm{p}(h_{[t+1} = j | h_{t]} = i) \mathrm{p}(h_{[t-d+1:t]} = i | h_{[t-d+1:t} = i) \mathrm{p}(h_{[t-d+1:t} = i, \mathbf{x}_{1:t}) \\ =&\sum_i \sum_d \mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_{t+1}) a_{i,j} \left(1 - \frac{D_i(d+1)}{D_i(d)}\right) \bar{\phi}_t(i,d) \end{aligned}$$

(II.11) Pour $d > 1$

$$\begin{aligned} &\mathrm{p}(h_{t+1} = j, d_{t+1} = d, \mathbf{x}_{1:t+1}) = \mathrm{p}(h_{[t-d+2:t+1} = j, \mathbf{x}_{1:t+1}) \\ =&\mathrm{p}(\mathbf{x}_{t+1} | h_{t+1} = j) \mathrm{p}(h_{[t-d+2:t+1} = j | h_{[t-d+2:t} = j) \mathrm{p}(h_{[t-d+2:t} = j, \mathbf{x}_{1:t}) \\ =&\mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_{t+1}) \left(\frac{D_i(d)}{D_i(d-1)}\right) \bar{\phi}_t(j,d-1) \end{aligned}$$

(II.12) Découle de la normalisation de $\bar{\phi}_{t+1}$

(II.13)

$$\begin{aligned}
&\frac{1}{t+1}\mathbb{E}\left[\sum_{l=1}^{t+1} S(\mathbf{x}_l, h_l, d_l|h_{l-1}, d_{l-1})|h_{t+1}=j, d_{t+1}=1, \mathbf{x}_{1:t+1}; \boldsymbol{\theta}\right] \\
=&\frac{1}{t+1}\sum_i\sum_d \mathbb{E}\left[\sum_{l=1}^{t+1} S(\mathbf{x}_l, h_l, d_l|h_{l-1}, d_{l-1})|h_{t+1}=j, h_t=i, d_{t+1}=1, d_t=d, \mathbf{x}_{1:t+1}; \boldsymbol{\theta}\right] \\
&\qquad \times \mathrm{p}(h_t=i, d_t=d|h_{t+1}=j, d_{t+1}=1, \mathbf{x}_{1:t}) \\
=&\frac{1}{t+1}\sum_i\sum_d \left[S(\mathbf{x}_{t+1}, j, 1|i, d) + \mathbb{E}\left[\sum_{l=1}^{t} S(\mathbf{x}_l, h_l, d_l|h_{l-1}, d_{l-1})|h_t=i, d_t=d, \mathbf{x}_{1:t}; \boldsymbol{\theta}\right]\right] \\
&\qquad \times r_{t+1}(i, d|j) \\
=&\sum_i\sum_d \left[\frac{1}{t+1}S(\mathbf{x}_{t+1}, j, 1|i, d) + \left(1-\frac{1}{t+1}\right)\rho_t(i, d)\right] r_{t+1}(i, d|j)
\end{aligned}$$

(II.14) Pour $d > 1$, on a l'équivalence $h_{t+1}=j, d_{t+1}=d \Leftrightarrow h_{t+1}=j, d_{t+1}=d, h_t=j, d_t=d-1$. D'où

$$\begin{aligned}
&\frac{1}{t+1}\mathbb{E}\left[\sum_{l=1}^{t+1} S(\mathbf{x}_l, h_l, d_l|h_{l-1}, d_{l-1})|h_{t+1}=j, d_{t+1}=d, \mathbf{x}_{1:t+1}; \boldsymbol{\theta}\right] \\
=&\frac{1}{t+1}\left[S(\mathbf{x}_{t+1}, j, d|j, d-1) + \mathbb{E}\left[\sum_{l=1}^{t} S(\mathbf{x}_l, h_l, d_l|h_{l-1}, d_{l-1})|h_t=j, d_t=d-1, \mathbf{x}_{1:t}; \boldsymbol{\theta}\right]\right] \\
=&\left[\frac{1}{t+1}S(\mathbf{x}_{t+1}, j, d|j, d-1) + \left(1-\frac{1}{t+1}\right)\rho_t(j, d-1)\right]
\end{aligned}$$

(II.15)

$$\begin{aligned}
&\mathrm{p}(h_t=i, d_t=d, h_{t+1}=j, d_{t+1}=1|\mathbf{x}_{1:t}) \\
=&\mathrm{p}(h_{[t-d+1:t]}=i, h_{[t+1}=j|\mathbf{x}_{1:t}) \\
=&\mathrm{p}(h_{[t+1}=j|h_{t]=i})\mathrm{p}(h_{[t-d+1:t]}=i|h_{[t-d+1:t}=i)\mathrm{p}(h_{[t-d+1:t}=i|\mathbf{x}_{1:t}) \\
=&a_{i,j}\left(1-\frac{D_i(d+1)}{D_i(d)}\right)\phi_t(i, d)
\end{aligned}$$

où l'on obtient le résultat après normalisation.

□

2. Algorithme

L'algorithme en ligne pour HSMM s'effectue alors de la même manière que pour les HMM : à chaque instant $t+1$ on estime $\phi_{t+1}(i, d)$ et $\rho_{t+1}(i, d)$ à partir des valeurs à l'instant précédent et de la nouvelle observation à $\mathbf{x}_{t+1}$, en remplaçant le pas d'incrémentation $\frac{1}{t+1}$ par un pas d'incrémentation stochastique γ_{t+1}. On peut alors estimer $\hat{s}^{(t+1)}$ et en déduire les nouveaux paramètres. L'algorithme 4 décrit le processus (on suppose que les distributions de durée sont de famille exponentielle). On peut estimer la classe la plus probable pour $\mathbf{x}_t$ à chaque itération t avec $\arg\max_j \sum_d \phi_t(j, d)$

Calcul de ρ_t D'après l'écriture des composantes de la statistique $S(\mathbf{x}_t, h_t, d_t|h_{t-1}, d_{t-1})$ (II.9), explicitée en début de section, on déduit alors la mise à jour des composantes de ρ_t correspondantes.

Centroïde
$$\rho_{t+1}^k(j, 1) \leftarrow \gamma_{t+1}\delta_{j,k}[1, \mathbf{x}_{t+1}] + (1-\gamma_{t+1})\sum_i\sum_d \rho_t^k(i, d)r_{t+1}(i, d|j)$$
$$\rho_{t+1}^k(j, d) \leftarrow \gamma_{t+1}\delta_{j,k}[1, \mathbf{x}_{t+1}] + (1-\gamma_{t+1})\rho_t^k(j, d-1)$$

Algorithme 4 Espérance-Maximisation en ligne pour HSMM

Procédure HSMM EN LIGNE($\mathbf{x}_t$ à l'instant t, paramètre $\boldsymbol{\theta}^{(1)}$ initial, $\gamma : \mathbb{N}^* \mapsto [0,1[$, t_{min})

$\phi_1(i,1) \leftarrow \pi_i \mathrm{e}^{-d_\phi(\mathbf{x}_1, \boldsymbol{\mu}_i^{(1)})}$, **Normalise** $\phi_1(\cdot,1)$

$\phi_1(i,d) \leftarrow 0$ pour $d > 1$

$\rho_1(i,d) \leftarrow \mathbf{0}$

Pour $t \leftarrow 1, \ldots, T-1$ **faire**

Espérance:

$$
\begin{aligned}
r_{t+1}(i,d|j) &\leftarrow a_{i,j}\left(1 - \tfrac{D_i(d+1)}{D_i(d)}\right)\phi_t(i,d), \quad \textbf{Normalise } r_{t+1}(\cdot,\cdot|j) \\
\bar{\phi}_{t+1}(j,1) &\leftarrow \textstyle\sum_i \sum_d \mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_{t+1}) a_{i,j}\left(1 - \tfrac{D_i(d+1)}{D_i(d)}\right)\bar{\phi}_t(i,d) \\
\rho_{t+1}(j,1) &\leftarrow \textstyle\sum_i \sum_d \left(\gamma_{t+1} S(\mathbf{x}_{t+1}, j, 1|i,d) + (1-\gamma_{t+1})\,\rho_t(i,d)\right) r_{t+1}(i,d|j)
\end{aligned}
$$

Pour $d > 1$

$$
\begin{aligned}
\bar{\phi}_{t+1}(j,d) &\leftarrow \mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_{t+1}) \tfrac{D_j(d)}{D_j(d-1)} \bar{\phi}_t(j,d-1) \\
\rho_{t+1}(j,d) &\leftarrow \gamma_{t+1} S(\mathbf{x}_{t+1}, j, d|j, d-1) + (1-\gamma_{t+1})\,\rho_t(j,d-1)
\end{aligned}
$$

$$
\begin{aligned}
\phi_{t+1} &\leftarrow \bar{\phi}_{t+1}, \quad \textbf{Normalise } \phi_{t+1} \\
\hat{s}^{(t+1)} &\leftarrow \textstyle\sum_j \sum_d \phi_{t+1}(j,d)\rho_{t+1}(j,d)
\end{aligned}
$$

Maximisation:

Si $t > t_{min}$ **then**

$$
\begin{aligned}
\boldsymbol{\mu}_i^{(t+1)} &\leftarrow \hat{s}_{i,2}^{(t+1)} / \hat{s}_{i,1}^{(t+1)} \\
a_{i,j}^{(t+1)} &\leftarrow \hat{s}_{i,j}^{(t+1)} / \hat{s}_{i,1}^{(t+1)} \\
\boldsymbol{\mu}_{D,i}^{t+1} &\leftarrow \textstyle\sum_d \hat{s}_{i,d}^{(t+1)} d / \sum_d \hat{s}_{i,d}^{(t+1)}
\end{aligned}
$$

Fin Si

Fin Pour

Retourne $\boldsymbol{\theta}^{(T)}, \phi_t(j,d)\ \forall t, j, d$

Fin Procédure

Transitions

$$
\begin{aligned}
\rho_{t+1}^{k,k'}(j,1) &\leftarrow \gamma_{t+1} \sum_d \delta_{j,k'} r_{t+1}(k,d|j) + (1-\gamma_{t+1}) \sum_i \sum_d \rho_t^{k,k'}(i,d) r(i,d|j) \\
\rho_{t+1}^{k,k'}(j,d) &\leftarrow (1-\gamma_{t+1}) \rho^{k,k'}(j,d-1)
\end{aligned}
$$

Durée

$$
\begin{aligned}
\rho_{t+1}^{k,d'}(j,1) &\leftarrow \gamma_{t+1} r_{t+1}(k,d'|j) + (1-\gamma_{t+1}) \sum_i \sum_d \rho_t^{k,d'}(i,d) r_{t+1}(i,d|j) \\
\rho_{t+1}^{k,d'}(j,d) &\leftarrow (1-\gamma_{t+1}) \rho_t^{k,d'}(j,d-1)
\end{aligned}
$$

Complexité On en déduit alors la complexité d'une itération, qui est, comme pour l'algorithme HMM en ligne, la complexité du calcul de ρ_t .La complexité spatiale est $O(K^2Dn + K^3D + K^2D^2)$, où n est la dimension des données, et la complexité temporelle est $O(K^3Dn + K^4D + K^3D^2)$.

Exemple II.1 Émissions gaussiennes, durée de Poisson

On décrit ici concrètement les calculs effectués dans le cas d'un modèle d'émission gaussien, ainsi qu'un modèle de durée suivant la loi de Poisson. En se référant au tableau I.2, on en déduit la statistique S de la loi exponentielle jointe utilisée pour le calcul en ligne des valeurs. Pour une loi gaussienne $\mathcal{N}(\mathbf{m}, \boldsymbol{\Sigma})$, la statistique exhaustive utilisée pour le calcul de la densité (I.6) est $[\mathbf{x}, \mathbf{x}\mathbf{x}^{\mathrm{T}}]$

pour une v.a. $\mathbf{x}$. Les paramètres d'espérance que l'on calcule lors de l'apprentissage sont donc $\boldsymbol{\mu} = [\mathbf{m}, \boldsymbol{\Sigma} + \mathbf{m}]$. On en déduit la statistique de la loi jointe $S_i(\mathbf{x}_t, h_t) = [\delta_{i,h_t}, \delta_{i,h_t}\mathbf{x}_t, \delta_{i,h_t}\mathbf{x}\mathbf{x}^{\mathrm{T}}]$.

Précision alors la formule de mise à jour de la valeur ρ_t, décrite dans l'algorithme 4, ainsi que celle du paramètre correspondant, et ce pour chaque paramètre du modèle $\boldsymbol{\theta}$,

Centroïdes On a vu que $S_k(\mathbf{x}_t, h_t) = [\delta_{k,h_t}, \delta_{k,h_t}\mathbf{x}_t, \delta_{k,h_t}\mathbf{x}\mathbf{x}^{\mathrm{T}}]$. L'élément de ρ_{t+1} correspondant à cette statistique, que nous écrirons ρ^k_{t+1} est donc mis à jour de la manière suivante :

$$\rho^k_{t+1}(j,1) \leftarrow \gamma_{t+1}\delta_{k,j}[1, \mathbf{x}_{t+1}, \mathbf{x}_{t+1}\mathbf{x}^{\mathrm{T}}_{t+1}] + (1-\gamma_{t+1}) \sum_i \sum_d \rho^k_t(i,d) r_{t+1}(i,d|j)$$

$$\rho^k_{t+1}(j,d) \leftarrow \gamma_{t+1}\delta_{k,j}[1, \mathbf{x}_{t+1}, \mathbf{x}_{t+1}\mathbf{x}^{\mathrm{T}}_{t+1}] + (1-\gamma_{t+1})\rho^k_t(j,d-1)$$

Le « vecteur » $\hat{s}^{(t+1)}$ obtenu a donc trois composantes. L'apprentissage des paramètres $\mathbf{m}$ et $\boldsymbol{\Sigma}$ se fait donc de la manière suivante.

$$\mathbf{m}^{(t+1)}_i \leftarrow \hat{s}^{(t+1)}_{i,2} / \hat{s}^{(t+1)}_{i,1}$$

$$\boldsymbol{\Sigma}^{(t+1)}_i \leftarrow (\hat{s}^{((t+1))}_{i,3} / \hat{s}^{(t+1)}_{i,1}) - \mathbf{m}^{(t+1)}$$

Durée Pour la loi de poisson $\mathcal{P}(\lambda)$, le paramètre d'espérance vaut $\boldsymbol{\mu}_D = \lambda$. Comme la loi de Poisson est de famille exponentielle, on en déduit donc la mise à jour du paramètre λ_i de la classe i :

$$\lambda^{(t+1)}_i \leftarrow \sum_d \hat{s}^{(t+1)}_{i,d} d / \sum_d \hat{s}^{(t+1)}_{i,d}$$

4. Algorithme incrémental, modèle de mélange

L'algorithme en ligne présenté ci-dessus reste stable, et sa convergence pour tous les modèles peut-être démontrée (Cappé 2011 ; Cappé et Moulines 2009). Cependant on remarquera que dans le cas des HMM et HSMM, la mise en place de l'algorithme nécessite le calcul des valeurs ϕ_t et ρ_t, avec une complexité élevée. Comme on le verra, le temps de calcul peut alors être très élevé pour un nombre de classes à détecter très important, ce qui peut être problématique notamment si on cherche à faire du temps réel. On va alors proposer un autre algorithme de partitionnement en ligne, cette fois-ci qualifié de *incrémental*, basé sur une approche introduite par Neal et al. (1998).

L'idée derrière cet algorithme incrémental et de revoir la théorie sur laquelle s'appuie l'algorithme EM, présenté en section I-4.2 (p.13). En effet la partie espérance consiste à trouver la meilleure borne inférieure $B(\boldsymbol{\theta}, \boldsymbol{\theta}^{(\tau)})$ à $l(\boldsymbol{\theta}) = \ln(\mathrm{p}(\mathcal{X}; \boldsymbol{\theta}))$. On avait trouvée que cette borne était $\sum_{\mathbf{h}} q^{(\tau)}(\mathbf{h}) \ln\left(\mathrm{p}(\mathcal{X}, \mathbf{h}) / q^{(\tau)}(\mathbf{h})\right)$, maximal pour $q^{(\tau)}(\mathbf{h}) = \mathrm{p}(\mathbf{h}|\mathcal{X}; \boldsymbol{\theta}^{(\tau)})$. Le problème est que pour calculer $q^{(\tau)}$, il est nécessaire de connaître l'ensemble des données, ce qui n'est pas la cas dans une situation d'apprentissage en ligne. En remarquant que $\mathrm{p}(\mathbf{h}|\mathcal{X}) = \prod_i \mathrm{p}(h_i|\mathbf{x}_i)$. On peut alors se limiter à q respectant cette propriété, *i.e.* $q^{(n)}(\mathbf{h}) = \prod_{i=1}^n q^{(n)}_i(h_i)$ avec $\sum_{k=1}^K q^{(n)}_i(k) = 1$. La borne inférieure devient alors

$$\begin{aligned}
B(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) &= \sum_{\mathbf{h}} \left(\prod_{j=1}^n q^{(n)}_i(h_j) \right) \sum_{i=1}^n \ln \left(\frac{\mathrm{p}(\mathbf{x}_i, h_i, \boldsymbol{\theta})}{q^{(n)}_i(h_i)} \right) \\
&= \sum_{i=1}^n \sum_{h_1=1}^K q^{(n)}_1(h_1) \dots \sum_{h_i=1}^K q^{(n)}_i(h_i) \ln \left(\frac{\mathrm{p}(\mathbf{x}_i, h_i, \boldsymbol{\theta})}{q_i(h_i)} \right) \dots \sum_{h_n=1}^K q^{(n)}_n(h_n) \\
&= \sum_{i=1}^n \sum_{h_i=1}^K q_i(h_i)^{(n)} \ln \left(\frac{\mathrm{p}(\mathbf{x}_i, h_i, \boldsymbol{\theta})}{q_i(h_i)} \right)
\end{aligned}$$

L'algorithme incrémental consistera donc à chaque étape (n) à choisir $q_i^{(n)}$ maximisant cette borne inférieure étant donné les paramètres $\boldsymbol{\theta}^{(i)}$: d'après ce qu'on a vu, on choisit alors $q_i^{(n)} = \mathrm{p}(h_i|\mathbf{x}_i;\boldsymbol{\theta}^{(n)})$. Puis on cherchera les nouveaux paramètres maximisant $B(\boldsymbol{\theta},\boldsymbol{\theta}^{(i)})/n$. Ce qui amène à

$$\boldsymbol{\theta}^{(n+1)} = \arg\max_{\boldsymbol{\theta}} \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\Big[\ln(\mathrm{p}(\mathbf{x}_i,h_i,\boldsymbol{\theta}))|\mathbf{x}_i;\boldsymbol{\theta}^{(n)}\Big]$$

Ce qui pour une loi jointe de famille exponentielle, telle qu'on l'a vu en section II-1.1, donne

$$\begin{aligned}\boldsymbol{\theta}^{(n+1)} &= \arg\max_{\boldsymbol{\theta}} \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\Big[\langle S(\mathbf{x}_i,h_i),\phi(\boldsymbol{\theta})\rangle\,|\mathbf{x}_i;\boldsymbol{\theta}^{(n)}\Big]\\ &= \arg\max_{\boldsymbol{\theta}} \Big\langle \hat{s}^{(n)},\phi(\boldsymbol{\theta})\Big\rangle\end{aligned}$$

On peut alors calculer les nouveaux paramètres avec S^n, comme on l'a vu précédemment.

Si on effectue le calcul de l'espérance de la statistique de manière incrémentale en utilisant une approximation stochastique, on obtient une expression similaire à celle de l'algorithme en ligne de Cappé et Moulines, décrit en section II-1.2. Si l'algorithme final est sensiblement, le raisonnement qui nous a amenés au résultat est lui différent. Pour les modèles hmm et hsmm, on va donc procéder de la même manière en décomposant la valeur $q(\mathbf{h})$, permettant une écriture en ligne de l'algorithme em appliquées à ces modèles.

5. HMM incrémental

L'algorithme em incrémental pour les hmm suit donc le même raisonnement (Bietti et al. 2015). Ici les observations ne sont plus indépendantes, la loi à priori se décompose de la manière suivante $\mathrm{p}(\mathbf{h}|\mathcal{X}) = \mathrm{p}(h_1|\mathbf{x}_1)\prod_{t=2}^{T}\mathrm{p}(h_t|h_{t-1},\mathbf{x}_t)$. On restreint donc à q prenant la forme $q(h_{1:T}) = q_1(h_1)\prod_{t\geq 2} q_t(h_t|h_{t-1})$ avec $\sum_j q(j|i) = 1$. En notant $\phi_1(h_1) = q_1(h_1)$ et $\phi_t(h_t) = \sum_{h_1,\dots,h_{t-1}} q_1(h_1)q_2(h_2|h_1)\dots q$ $\sum_{h_{t-1}}\phi_{t-1}(h_{t-1})q_t(h_t)$. On peut alors redéfinir la borne inférieure en fonction de q et ϕ (en prenant la convention $q_1(h_1|h_0) = q_1(h_1)$)

$$\begin{aligned}\sum_{\mathbf{h}} q(\mathbf{h})\ln\left(\frac{\mathrm{p}(\mathcal{X},\mathbf{h};\boldsymbol{\theta})}{q(\mathbf{h})}\right) &= \sum_{t=1}^{T}\sum_{\mathbf{h}} q(\mathbf{h})\ln\left(\frac{\mathrm{p}(\mathbf{x}_t,h_t|h_{t-1};\boldsymbol{\theta})}{q_t(h_t|h_{t-1})}\right)\\ &= \sum_{t=1}^{t}\sum_{h_t}\sum_{h_{t-1}}\phi_{t-1}(h_{t-1})q_t(h_t|h_{t-1})\ln\left(\frac{\mathrm{p}(\mathbf{x}_t,h_t|h_{t-1};\boldsymbol{\theta})}{q_t(h_t|h_{t-1})}\right)\end{aligned}$$

Considérons alors un instant T donnée, on cherche pour $\phi_{t-1}(h_{t-1})$ et $q_t(h_l)$ fixés avec $t < T$, quelle fonction $q_T(h_T|h_{T-1})$ maximise cette borne inférieure.

$$G(q_T) = \sum_{t=1}^{T}\sum_{h_t}\sum_{h_{t-1}}\phi_{t-1}(h_{t-1})q_t(h_t|h_{t-1})\ln\left(\frac{\mathrm{p}(\mathbf{x}_t,h_t|h_{t-1};\boldsymbol{\theta})}{q_t(h_t|h_{t-1})}\right) + \lambda\left[1 - \sum_{h_T} q_T(h_T|h_{T-1})\right]$$

$$\frac{\delta G}{\delta h_T(j|i)} = \phi_{T-1}(i)[\ln(\mathrm{p}(\mathbf{x}_T,j|i)) - \ln(q_T(j|i)) - 1] - \lambda = 0$$

En divisant la dérivée du lagrangien par $\phi_{T-1}(i)$ et en sommant sur j, puisque $\sum_j q_T(j|i) = 1$, on trouve que $\lambda + \phi_{T-1}(i) = \phi_{T-1}(i) \ln(\sum_j \mathrm{p}(\mathbf{x}_T, j|i)) = \phi_{T-1}(i) \ln(\mathrm{p}(\mathbf{x}_T|h_{T-1} = i)$. D'où

$$\begin{aligned} q_T(j|i) &= \frac{\mathrm{p}(\mathbf{x}_T, h_T = j|h_{T-1} = i)}{\mathrm{p}(\mathbf{x}_T|h_{T-1} = i)} = \mathrm{p}(h_T = j|h_{T-1} = i, \mathbf{x}_T) \\ &= \frac{\mathrm{p}_{(\psi, \boldsymbol{\eta}_j^{(T-1)})}(\mathbf{x}_T) a_{i,j}^{(T-1)}}{\sum_k \mathrm{p}_{(\psi, \boldsymbol{\eta}_k^{(T-1)})}(\mathbf{x}_T) a_{i,k}^{(t-1)}} \end{aligned}$$

L'étape de maximisation se déduit alors de la même manière que pour le modèle de mélange. La maximisation de la borne inférieure en fonction de $\boldsymbol{\theta}$, en considérant toujours que l'on utilise des émissions de famille exponentielle, donne

$$\begin{aligned} &\arg\max_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t=1}^{T} \sum_{h_t} \sum_{h_{t-1}} \phi_{t-1}(h_{t-1}) q_t(h_t|h_{t-1}) \ln(\mathrm{p}(\mathbf{x}_t, h_t|h_{t-1}; \boldsymbol{\theta})) \\ &\arg\max_{\boldsymbol{\theta}} \langle \hat{s}, \phi(\boldsymbol{\theta}) \rangle \end{aligned}$$

Où $\hat{s} = \frac{1}{T} \sum_t \sum_i \sum_j \phi_{t-1}(i) q_t(j) S(\mathbf{x}_t, i, j)$. S étant la statistique exhaustive définie en section II-2.

On en déduit alors l'incrémentation de cette statistique. L'algorithme 5 décrit le fonctionnement de l'algorithme HMM incrémental.

Algorithme 5 Espérance-Maximisation incrémental pour HMM

Procédure HMM INCRÉMENTAL($\mathbf{x}_t$ à l'instant t, paramètre $\boldsymbol{\theta}^{(1)}$ initial, $\gamma : \mathbb{N}^* \mapsto [0, 1[$, t_{min})

$\phi_1(i) \leftarrow \pi_i \mathrm{e}^{-d_\phi(\mathbf{x}_1, \boldsymbol{\mu}_i^{(1)})} \quad \forall i \in [\![1; K]\!]$, **Normalise** ϕ_1

$\hat{s}^{(1)} \leftarrow \sum_i \phi_1(i)$

Pour $t \leftarrow 1, \ldots, T-1$ **faire**

Espérance:

$$\begin{aligned} q_{t+1}(j|i) &\leftarrow \mathrm{p}_{(\psi, \boldsymbol{\eta}_j^{(t)})}(\mathbf{x}_t) a_{i,j}^{(t)} \quad \forall i \in [\![1; K]\!], \quad \textbf{Normalise } q_{t+1}(\cdot|i) \\ \phi_{t+1}(j) &\leftarrow \sum_i \phi_t(i) q_t(j|i) \\ \hat{s}^{(t+1)} &\leftarrow (1 - \gamma_{t+1}) \hat{s}^{(t)} + \gamma_{t+1} \sum_i \sum_j \phi_t(i) q_{t+1}(j|i) S(\mathbf{x}_{t+1}, j|i) \end{aligned}$$

Maximisation:

Si $t > t_{min}$ **then**

$\boldsymbol{\mu}_i^{(t+1)} \leftarrow \hat{s}_{i,2}^{(t+1)} / \hat{s}_{i,1}^{(t+1)}$

$a_{i,j}^{(t+1)} \leftarrow \hat{s}_{i,j}^{(t+1)} / \hat{s}_{i,1}^{(t+1)}$

Fin Si

Fin Pour

Retourne $\boldsymbol{\theta}^{(T)}, \phi_t(j) \; \forall t, j$

Fin Procédure

Calcul de $\hat{s}^{(t)}$ Comme pour l'algorithme en ligne, on peut expliciter la mise à jour des composantes de $\hat{s}^{(t)}$, à partir des composantes de $S(\mathbf{x}_t, h_t|h_{t-1})$. On en déduit alors les mises à jour :

$$\begin{aligned} \hat{s}_k^{(t+1)} &\leftarrow (1 - \gamma_{t+1}) \hat{s}_k^{(t)} + \gamma_{t+1} \phi_{t+1}(k) [1, \mathbf{x}_{t+1}] \\ \hat{s}_{k,k'}^{(t+1)} &\leftarrow (1 - \gamma_{t+1}) \hat{s}_{k,k'}^{(t)} + \gamma_{t+1} \phi_t(k) q_{t+1}(k'|k) \end{aligned}$$

La première mise à jour étant due à l'égalité $\phi_{t+1}(k) = \sum_i \phi_t(i) q_{t+1}(k|i)$.

Complexité On en déduit alors une complexité spatiale et temporelle en $O(nK + K^2)$. L'ordre de grandeur de la complexité et donc réduit par rapport à la version *en ligne*.

6. HSMM incrémental

Finalement, on présente l'algorithme d'apprentissage incrémental pour les HSMM. Tout comme pour l'algorithme en ligne, il faut maintenant considérer une seconde variable caché d_t (indiquant toujours la taille de la séquence qui émet l'observation $\mathbf{x}_t$ à l'instant t). On a alors l'inégalité $\ln(\mathrm{p}(\mathcal{X};\boldsymbol{\theta})) \geq \sum_{\mathbf{h}} \sum_{\mathbf{d}} \mathrm{p}(\mathbf{h},\mathbf{d},\mathcal{X};\boldsymbol{\theta})$. Si décompose $\mathrm{p}(\mathbf{h},\mathbf{d}|\mathcal{X}) = \mathrm{p}(h_1,d_1|\mathbf{x}_1)\prod_{t=2}^{T}\mathrm{p}(h_t,d_t|h_{t-1},d_{t-1},\mathbf{x}_t)$, on se restreint donc à $q(\mathbf{h},\mathbf{d})$ de la même forme se factorisant par les termes $q_t(h_t,d_t|h_{t-1},d_{t-1})$. On redéfini la borne inférieure de façon similaire au cas HMM

$$
\begin{aligned}
&\sum_{\mathbf{h}}\sum_{\mathbf{d}} q(\mathbf{h},\mathbf{d}) \ln\left(\frac{\mathrm{p}(\mathcal{X},\mathbf{h},\mathbf{d})}{q(\mathbf{h},\mathbf{d})}\right)\\
=&\sum_{t=1}^{T}\sum_{\mathbf{h}}\sum_{\mathbf{d}} q(\mathbf{h},\mathbf{d}) \ln\left(\frac{\mathrm{p}(\mathbf{x}_t,h_t,d_t|h_{t-1},d_{t-1})}{q(h_t,d_t|h_{t-1},d_{t-1})}\right)\\
=&\sum_{t=1}^{T}\sum_{h_t,d_t}\sum_{h_{t-1}d_{t-1}} \phi_t(h_{t-1},d_{t-1})q_t(h_t,d_t|h_{t-1},d_{t-1}) \ln\left(\frac{\mathrm{p}(\mathbf{x}_t,h_t,d_t|h_{t-1},d_{t-1})}{q(h_t,d_t|h_{t-1},d_{t-1})}\right)
\end{aligned}
$$

Avec toujours $\phi_t(h_t,d_t) = \sum_{h_{t-1}}\sum_{d_{t-1}} \phi_{t-1}(h_{t-1},d_{t-1})q_t(h_t,d_t|h_{t-1},d_{t-1})$

Si on fixe alors les valeurs utiles pour t < T, et que l'on cherche trouver q_T maximisant cette borne inférieure, on trouve en suivant le même raisonnement que la solution est

$$q_T(h_T,d_T|h_{T-1},d_{T-1}) = \mathrm{p}(h_T,d_T|h_{T-1},d_{T-1},\mathbf{x}_T)$$

Soit une séquence de taille $d_{t-1} = d-1$ à un instant $t-1$. À l'instant suivant t, soit la séquence continue, alors $h_t = h_{t-1}$ et $d_t = d$, soit il y a changement d'état et transition, et $d_t = 1$. On en conclut donc que pour d_{t-1} et d_t ne respectant pas cette condition, la valeur q_t est nulle. On en déduit alors les expressions des valeurs utiles.

Proposition II.3 *Soit q_t et ϕ_t définis ci-dessus, et $\bar{q}_t(h_t,d_t|h_{t-1},d_{t-1}) = \mathrm{p}(\mathbf{x}_t,h_t,d_t|h_{t-1},d_{t-1})$. Alors*

$$\bar{q}_t(j,1|i,d) = \mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_t)a_{i,j}\left(1-\frac{D_i(d+1)}{D_i(d)}\right) \tag{II.16}$$

$$\bar{q}_t(i,d+1|i,d) = \mathrm{p}_{(\psi,\boldsymbol{\eta}_i)}(\mathbf{x}_t)\left(1-\frac{D_i(d+1)}{D_i(d)}\right) \tag{II.17}$$

$$\bar{q}_t(j,d'|i,d) = 0 \quad \text{sinon}$$

$$q_t(j,d'|i,d) = \frac{\bar{q}_t(j,d'|i,d)}{\bar{q}_t(i,d+1|i,d) + \sum_k \bar{q}_t(k,d+1|i,d)} \tag{II.18}$$

$$\phi_t(j,1) = \sum_i\sum_d \phi_{t-1}(i,d)q_t(j,1|i,d) \tag{II.19}$$

$$\phi_t(i,d) = \phi_{t-1}(i,d-1)q_t(i,d|i,d-1) \tag{II.20}$$

Preuve :

(II.16)
$$\begin{aligned}&\mathrm{p}(\mathbf{x}_t, h_t = j, d_t = 1 | h_{t-1} = i, d_{t-1} = d) = \mathrm{p}(\mathbf{x}_t, h_{[t} = j | h_{[t-d:t-1} = i)\\ =&\mathrm{p}(\mathbf{x}_t | h_t = j)\mathrm{p}(h_{[t} = j | h_{t-1]} = i)\mathrm{p}(h_{[t-d:t-1]} = i | h_{[t-d:t-1} = i)\\ =&\mathrm{p}_{(\psi,\boldsymbol{\eta}_j)}(\mathbf{x}_t) a_{i,j} \left(1 - \frac{D_i(d+1)}{D_i(d)}\right)\end{aligned}$$

(II.17)
$$\begin{aligned}&\mathrm{p}(\mathbf{x}_t, h_t = i, d_t = d+1 | h_{t-1} = i, d_{t-1} = d) = \mathrm{p}(\mathbf{x}_t, h_{[t-d:t} = i | h_{[t-d:t-1} = i)\\ =&\mathrm{p}(\mathbf{x}_t | h_t = i)\mathrm{p}(h_{[t-d:t} = i | h_{[t-d:t-1} = i)\\ =&\mathrm{p}_{(\psi,\boldsymbol{\eta}_i)}(\mathbf{x}_t) \left(1 - \frac{D_i(d+1)}{D_i(d)}\right)\end{aligned}$$

(II.18) On obtient $\mathrm{p}(\mathbf{x}_t)$ en sommant $\bar{q}_t(j, d'|i, d)$ pour tout j et tout d' tel que q_t est non nul.
(II.19), (II.20) Découle de la définition de ϕ_t et des conditions pour que q_t soit non nul.

□

L'espérance de la statistique a pour expression

$$\begin{aligned}\hat{s} &= \frac{1}{T}\sum_{t=1}^{T}\sum_i\sum_j\sum_{d'}\sum_d \phi_{t-1}(i,d) q_t(j,d'|i,d)\\ &= \frac{1}{T}\sum_{t=1}^{T}\sum_i\sum_d \phi_{t-1}(i,d)\left(q_t(i,d+1|i,d) + \sum_j q_t(j,1|i,d)\right)\end{aligned}$$

On en déduit alors l'algorithme 6

Calcul de $\hat{s}^{(t)}$ Explicitons la mise à jour de la valeur $\hat{s}^{(t)}$ en fonction de ses composantes

$$\begin{aligned}\hat{s}_j^{(t+1)} &\leftarrow (1-\gamma_{t+1})\hat{s}_j^{(t)} + \gamma_{t+1}\sum_d \phi_{t+1}(j,d)[1, \mathbf{x}_{t+1}]\\ \hat{s}_{i,j}^{(t+1)} &\leftarrow (1-\gamma_{t+1})\hat{s}_{i,j}^{(t)} + \gamma_{t+1}\sum_d \phi(i,d) q_{t+1}(j,1|i,d)\\ \hat{s}_{i,d}^{(t+1)} &\leftarrow (1-\gamma_{t+1})\hat{s}_{i,d}^{(t)} + \gamma_{t+1}\sum_j \phi_t(i,d) q_{t+1}(j,1|i,d)\end{aligned}$$

Complexité On en déduit une complexité spatiale en $O(nK + K^2 + KD)$ et une complexité temporelle en $O(nKD + K^2D)$. La complexité de l'algorithme HSMM *en ligne* à un ordre de grandeur bien plus élevé.

Exemple II.2 Émissions multinomiales, durée binomiales négatives

L'espérance de la statistique pour une loi multinomiale de paramètres $[p_1, \ldots, p_d]$ et N est alors $\boldsymbol{\mu} = [Np_1, \ldots, Np_d]$ (exemple I.4). On en déduit

$$\boldsymbol{\mu}_i^{(t+1)} \leftarrow \hat{s}_{i,2}^{(t+1)} / N\hat{s}_{i,1}^{(t+1)}$$

En considérant uniquement le paramètre p de la loi binomiale négative $\mathcal{N}(r,p)$, celle-ci est de famille exponentielle (voir section I-6.2), et son espérance étant $\boldsymbol{\mu}_D = r(1-p)/p$, on peut alors calculer la mise à jour avec

$$p_i^{(t+1)} \leftarrow \frac{r}{r + \frac{\sum_d \hat{s}_{i,d}^{(t+1)} d}{\sum_d \hat{s}_{i,d}^{(t+1)}}}$$

Algorithme 6 Espérance-Maximisation incrémental pour HSMM

Procédure HSMM INCRÉMENTAL($\mathbf{x}_t$ à l'instant t, paramètre $\boldsymbol{\theta}^{(1)}$ initial, $\gamma : \mathbb{N}^* \mapsto [0, 1[$, t_{min})

$\phi_1(i,1) \leftarrow \pi_i \mathrm{e}^{-d_\phi(\mathbf{x}_1, \boldsymbol{\mu}_i^{(1)})}$, **Normalise** $\phi_1(\cdot, 1)$

$\phi_1(i) \leftarrow 0$, pour $d > 1$

$\hat{s}^{(1)} \leftarrow \sum_i \phi_1(i)$

Pour $t \leftarrow 1, \ldots, T-1$ **faire**

Espérance:

$$q_{t+1}(j, 1|i, d) \leftarrow \mathrm{p}_{(\psi, \boldsymbol{\eta}_j^{(t)})}(\mathbf{x}_{t+1}) a_{i,j}^{(t)} \left(1 - \frac{D_i^{(t)}(d+1)}{D_i^{(t)}(d)}\right)$$

$$q_{t+1}(j, d+1|j, d) \leftarrow \mathrm{p}_{(\psi, \boldsymbol{\eta}_j^{(t)})}(\mathbf{x}_{t+1} t) \left(1 - \frac{D_j^{(t)}(d+1)}{D_j^{(t)}(d)}\right)$$

Normalise $q_{t+1}(\cdot, \cdots | i, d)$

$$\phi_{t+1}(j, 1) \leftarrow \sum_i \sum_d \phi_t(i, d) q_t(j, 1|i, d)$$

$$\phi_{t+1}(j, d+1) \leftarrow \phi_t(j, d) q_t(j, d+1|i, d)$$

$$\hat{s}^{(t+1)} \leftarrow (1 - \gamma_{t+1}) \hat{s}^{(t)} + \gamma_{t+1} \sum_{i,j} \sum_{d,d'} \phi_t(i, d) q_{t+1}(j, d'|i, d) S(\mathbf{x}_{t+1}, j, d'|i, d)$$

Maximisation:

Si $t > t_{min}$ **then**

$\boldsymbol{\mu}_i^{(t+1)} \leftarrow \hat{s}_{i,2}^{(t+1)} / \hat{s}_{i,1}^{(t+1)}$

$a_{i,j}^{(t+1)} \leftarrow \hat{s}_{i,j}^{(t+1)} / \hat{s}_{i,1}^{(t+1)}$

$\boldsymbol{\mu}_{D,i}^{(t+1)} \leftarrow \sum_d \hat{s}_{i,d}^{(t+1)} d / \hat{s}_{i,d}^{(t+1)}$

Fin Si

Fin Pour

Retourne $\boldsymbol{\theta}^{(T)}, \phi_t(j)\ \forall t, j$

Fin Procédure

Chapitre III

Résultats expérimentaux

Après avoir présenté durant les premiers chapitres des algorithmes de partitionnement, tout d'abord hors ligne, puis en ligne, nous allons ensuite présenter quelques résultats concernant de ces algorithmes. Tout d'abord nous montrerons comment les algorithmes réagissent face à des données synthétisées selon le modèle sur lequel ils reposent, et ensuite face à des exemples de signaux audio tirés de jeu de données. Ensuite nous évaluerons ces algorithmes ; nous introduirons le protocole utilisé à cette fin, puis nous présenterons les résultats, comparés à ceux de l'état de l'art.

Tous les algorithmes présentés ont été implémentés en utilisant le langage de programmation `python`, ainsi que la bibliothèque de calcul scientifique `numpy/scipy`. L'ensemble des graphiques sont générés à l'aide de la bibliothèque `matplotlib`.

1. Segmentation de données synthétiques

Afin de donner un aperçu de l'efficacité de ces algorithmes de partitionnement, nous allons tout d'abord présenter quelques résultats sur des exemples précis, synthétisés selon le modèle HSMM présenté en section I-6.1.

On génère les données dans l'espace $\mathbb{R}^2$ (afin de pouvoir les représenter par des points dans un plan). On construit ces observations selon $K = 4$ classes différentes, chacune ayant ses propres paramètres d'émission et de durée. On utilise la loi normale $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ pour les émissions, et la loi de Poisson $\mathcal{P}(\lambda)$ pour la durée des séquences. Les paramètres de chaque classe sont

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 1.5 \\ 2.0 \end{bmatrix}, \quad \boldsymbol{\mu}_3 = \begin{bmatrix} -2.0 \\ -1.2 \end{bmatrix}, \quad \boldsymbol{\mu}_4 = \begin{bmatrix} 4.0 \\ 3.0 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 2.0 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 0.5 & 0.2 \\ 0.2 & 1.2 \end{bmatrix}, \quad \boldsymbol{\Sigma}_3 = \begin{bmatrix} 1.1 & -0.65 \\ -0.65 & 1.0 \end{bmatrix}, \quad \boldsymbol{\Sigma}_4 = \begin{bmatrix} 0.87 & 0.0 \\ 0.0 & 0.87 \end{bmatrix}$$

$$\lambda_1 = 5, \quad \lambda_2 = 10, \quad \lambda_3 = 12, \quad \lambda_4 = 7$$

Nous utilisons les paramètres suivants pour la chaîne de Markov :

$$\mathbf{A} = \begin{bmatrix} 0.3 & 0.2 & 0.4 & 0.1 \\ 0.5 & 0.2 & 0.15 & 0.15 \\ 0.1 & 0.45 & 0.3 & 0.15 \\ 0.6 & 0.1 & 0.2 & 0.1 \end{bmatrix}, \quad \boldsymbol{\pi} = [0.3, 0.4, 0.2, 0.1]$$

On synthétise alors $T = 4000$ valeurs selon le processus suivant

1. On tire une classe i selon $\boldsymbol{\pi}$.
2. On tire une durée d selon $\mathcal{P}(\lambda_i)$.
3. On tire d observations dans $\mathbb{R}^2$ selon $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$.
4. On stocke les observations dans un vecteur $\mathbf{x}$, et le numéro de la classe dans un vecteur d'annotation.

5. On tire une nouvelle classe j selon $(\mathbf{A})_i$. On prend alors $i \leftarrow j$.
6. On réitère à l'étape 2, jusqu'à obtenir T observations.

La figure III.1 représente les données ainsi synthétisées sous forme de points dans un plan.

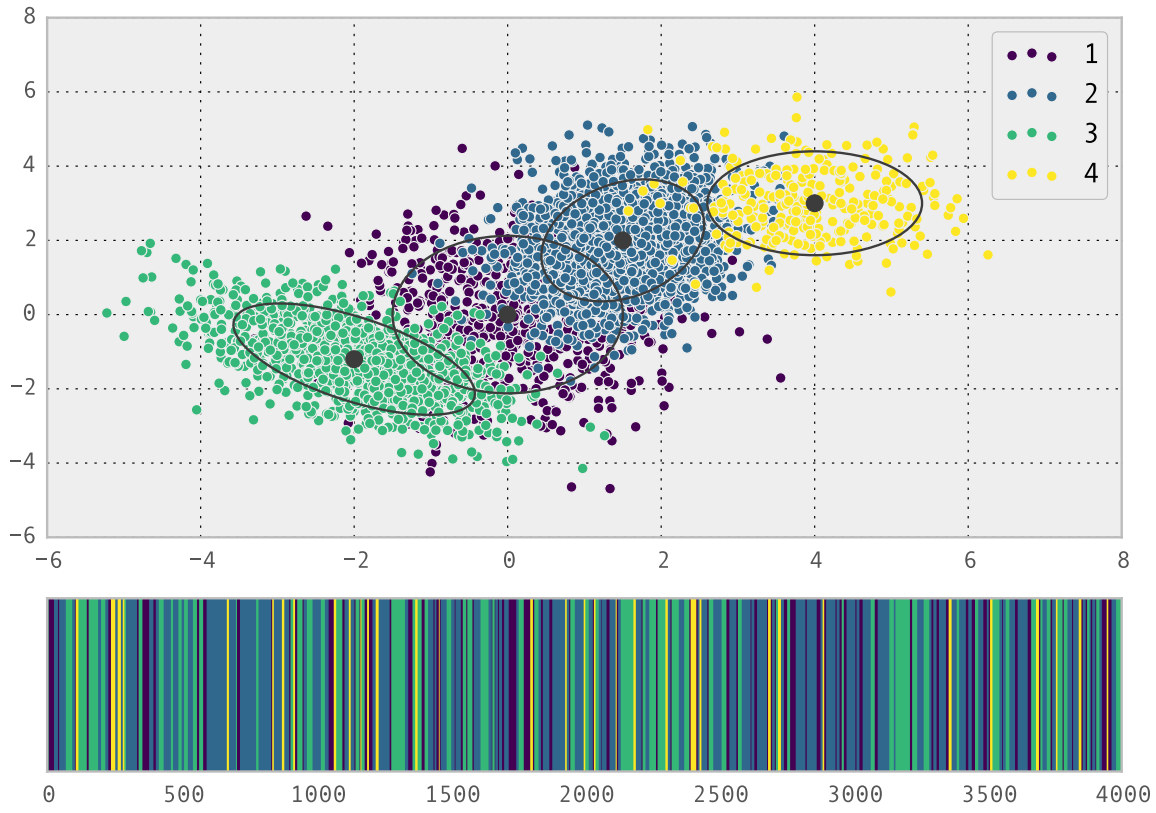


FIGURE III.1 – Données synthétisées

Haut Chaque point représente une donnée, la classe à laquelle appartient ce point étant déterminée par sa couleur. Les points noirs représentent les paramètres $\boldsymbol{\mu}$ utilisés par chaque classe. Les paramètres de covariance $\boldsymbol{\Sigma}$ sont représentés sous forme d'ellipses de confiance.

Bas Représentation temporelle de l'émission des données, chaque tranche de couleur représente une séquence émise par la classe correspondante

Nous allons alors pouvoir lancer les différents algorithmes sur les données non étiquetées. Le but du partitionnement sera alors de retrouver la classe à laquelle appartient chaque observation. On pourra de plus comparer les valeurs des paramètres HSMM estimées par l'algorithme avec celles utilisées pour la génération des données.

1. Algorithme hors-ligne

Réalisons tout d'abord un apprentissage avec l'algorithme EM hors-ligne pour HSMM, aussi appelé *batch* EM. On utilise bien sûr, comme pour la génération des données, la loi normale $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ comme modèle d'émission d'observations, et la loi de Poisson $\mathcal{P}(\lambda)$ comme modèle de durée des séquences. L'algorithme est exécuté en initialisant les paramètres $\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}$ grâce à l'algorithme EM pour modèle de mélange, lui-même initialisé par l'algorithme des K-Moyennes. La matrice de transition $\mathbf{A}$ est initialisée avec $a_{i,j} = 1/K \ \forall i, j \in [\![1; K]\!]$, et les distributions de durée sont initialisées avec $\lambda = 5$. L'algorithme d'apprentissage comprend 10 itérations.

L'initialisation par K-Moyennes permet de lancer l'algorithme avec des centroïdes déjà bien définis pour chaque classe. Le résultat final dépendant fortement de l'initialisation (*c.f.* la discussion sur le maximum local trouvé par EM, section I-4.2), une initialisation randomisée donnerait des résultats très variables.

La figure III.2 montre le résultat donné sous la même forme que précédemment, avec en plus la vérité-terrain (les séquences de classes originelles).

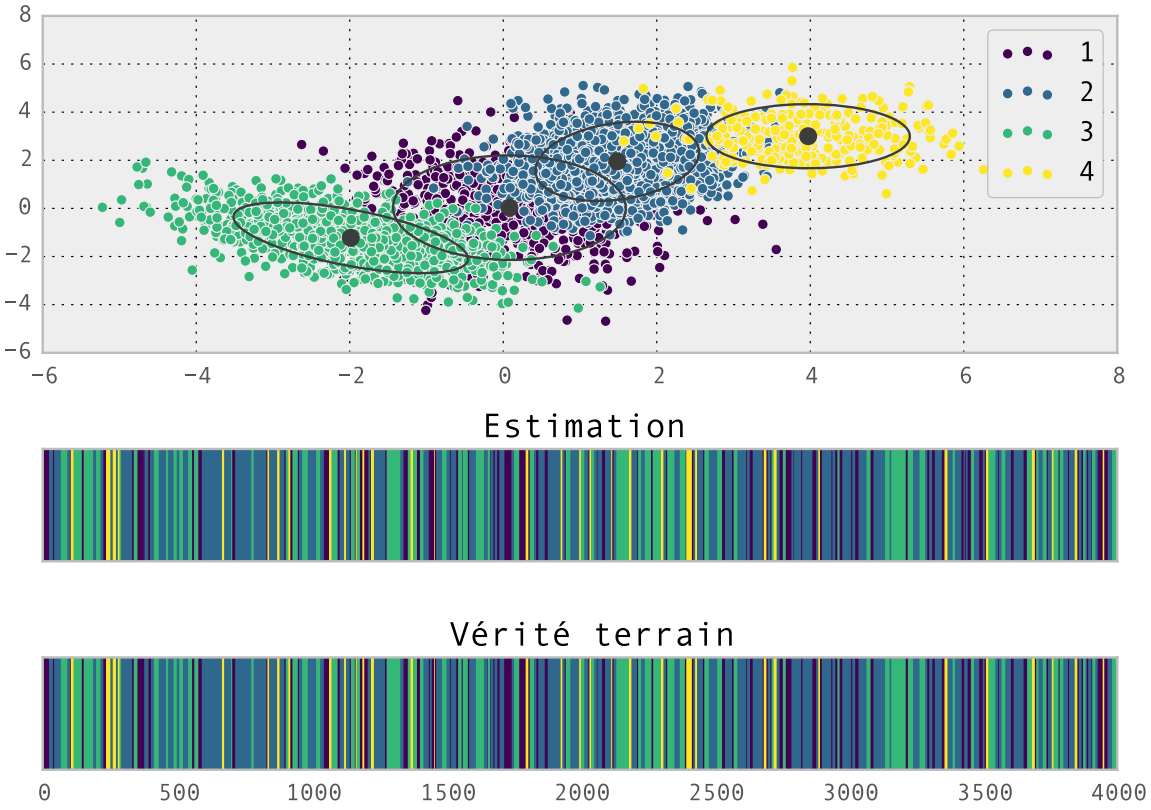


FIGURE III.2 – Partitionnement des données grâce à l'algorithme EM et comparaison avec la vérité-terrain

Étant donné que le modèle utilisé pour la génération des observations correspond parfaitement au modèle sur lequel repose l'algorithme, le partitionnement des données est quasi-sans-faute. Pour donner une estimation de la qualité du partitionnement, sa F-Mesure [1] vaut $98,5\%$. On peut également comparer les paramètres estimés à l'issue de l'apprentissage et les paramètres utilisés pour la synthèse [2]. On trouve

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 0.09 \\ 0.04 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 1.48 \\ 1.96 \end{bmatrix}, \quad \boldsymbol{\mu}_3 = \begin{bmatrix} -1.99 \\ -1.22 \end{bmatrix}, \quad \boldsymbol{\mu}_4 = \begin{bmatrix} 3.97 \\ 3.00 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 1.04 & 0.01 \\ 0.01 & 2.11 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 0.51 & 0.18 \\ 0.18 & 1.20 \end{bmatrix}, \quad \boldsymbol{\Sigma}_3 = \begin{bmatrix} 1.04 & -0.59 \\ -0.59 & 0.96 \end{bmatrix}, \quad \boldsymbol{\Sigma}_4 = \begin{bmatrix} 0.78 & 0.00 \\ 0.00 & 0.79 \end{bmatrix}$$

$$\lambda_1 = 5.92, \quad \lambda_2 = 8.24, \quad \lambda_3 = 5.16, \quad \lambda_4 = 9.07$$

1. La F-mesure est une métrique permettant d'évaluer l'efficacité du partitionnement étant donné la vérité-terrain : plus elle est élevée plus le partitionnement est réussi. On définira cette métrique et d'autres plus précisément en section III-3.2.
2. Les résultats sont arrondis à deux chiffres après la virgule pour faciliter la lecture.

Les paramètres markoviens trouvés sont

$$\mathbf{A} = \begin{bmatrix} 0.28 & 0.36 & 0.24 & 0.11 \\ 0.25 & 0.58 & 0.11 & 0.05 \\ 0.11 & 0.17 & 0.69 & 0.03 \\ 0.32 & 0.38 & 0.26 & 0.04 \end{bmatrix}, \ \boldsymbol{\pi} = [0.00, 0.00, 1.00, 0.00]$$

Ces valeurs sont plutôt cohérentes, à l'exception de π, étant donnée que notre apprentissage n'est basé que sur un seul jeu de données, on aboutit forcément à une classe qui est bien plus probable que les autres en début de séquence.

On peut aussi afficher l'évolution de la log-vraisemblance des données en fur et à mesure des itérations (figure III.3). On constate que l'algorithme converge très rapidement, en seulement quelques itérations.

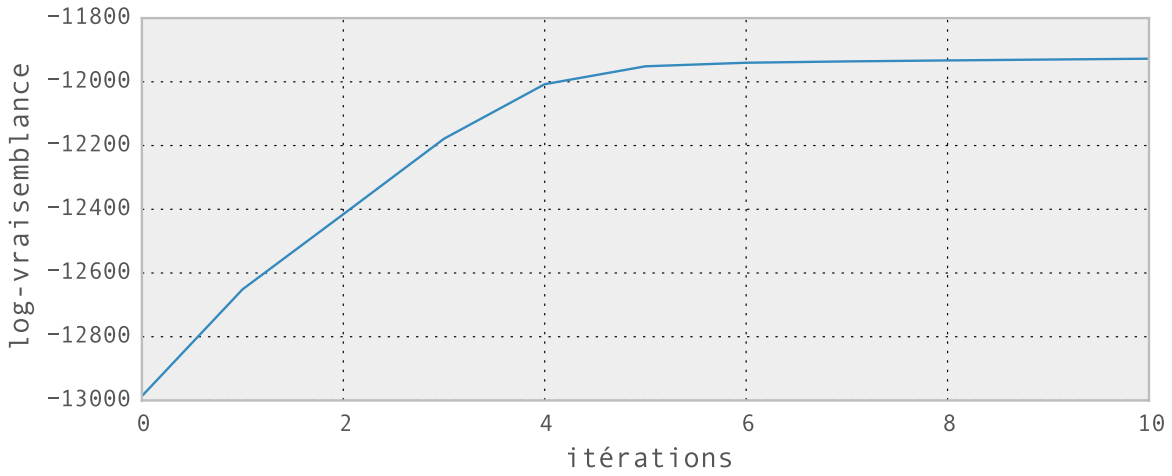


FIGURE III.3 – Log-Vraisemblance pour l'algorithme HSMM avec données synthétiques

Bien évidemment, cet exemple n'est absolument pas un cas pratique : les observations ont été générées exactement selon le modèle sur lequel est construit l'algorithme. Cependant, cela nous permet de vérifier que la théorie sur laquelle est basée la méthode Espérance-Maximisation est bien construite. On peut maintenant appliquer les algorithmes *en ligne* et *incrémental* à cette même séquence, afin de comparer au cas hors ligne.

2. Algorithmes en ligne

Dans le cas des algorithmes en ligne, l'initialisation ne peut pas se faire avec les K-Moyennes, ceci nécessiterait une connaissance de l'ensemble des données. On initialise alors en tirant quatre points selon une loi normale $\mathcal{N}([0,0], \sigma^2 = 1)$. On initialise ensuite $\boldsymbol{\Sigma}_k = \mathrm{I}_2$ où I_d est la matrice identité de dimension d, et $\pi_k = 1/K$. Les autres paramètres sont initialisés comme que précédemment. Le pas d'incrémentation est fixé à $\gamma_t = 1/t$.

Version en ligne

Commençons tout d'abord avec l'algorithme en ligne inspiré de CAPPÉ (2011). La figure III.4 montre le résultat du partitionnement avec l'algorithme en ligne. L'algorithme est, comme on peut s'y attendre, bien moins performant que le cas hors-ligne. On trouve pour ce partitionnement une F-mesure de $65,2\%$. On constate que les quelques premières observations sont médiocrement classées,

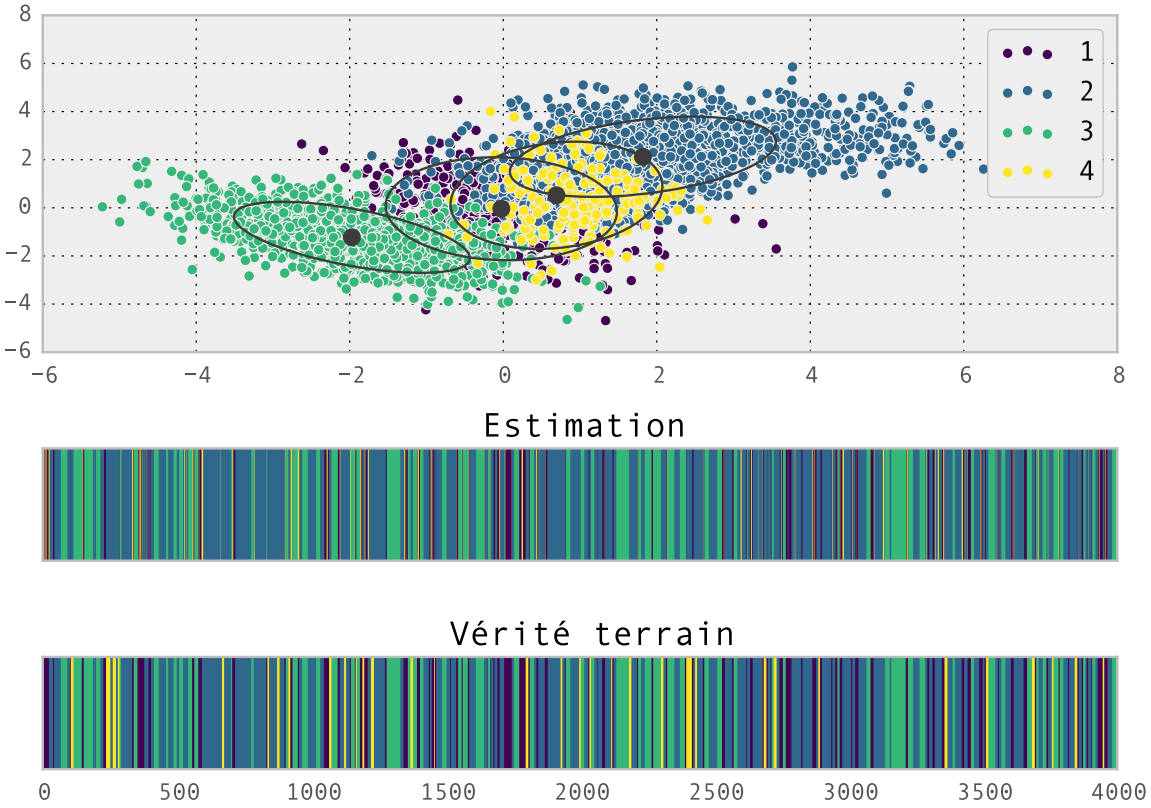


FIGURE III.4 – Partitionnement des données grâce à l'algorithme EM en ligne et comparaison avec la vérité-terrain

cela correspond au temps t_{min} avant que l'on commence à ré-estimer les paramètres (*c.f.* section II-2.2 p.30). Pour le reste, le résultat est plutôt correct, si ce n'est pour la classe 4, qui n'est presque pas détectée.

Les paramètres trouvés à l'issue de l'algorithme sont les suivants

$$\boldsymbol{\mu}_1 = \begin{bmatrix} -0.03 \\ -0.04 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 1.82 \\ 2.12 \end{bmatrix}, \quad \boldsymbol{\mu}_3 = \begin{bmatrix} -1.97 \\ -1.23 \end{bmatrix}, \quad \boldsymbol{\mu}_4 = \begin{bmatrix} 0.69 \\ 0.52 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 1.01 & -0.08 \\ -0.08 & 2.04 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 1.34 & 0.48 \\ 0.48 & 1.24 \end{bmatrix}, \quad \boldsymbol{\Sigma}_3 = \begin{bmatrix} 1.05 & -0.59 \\ -0.59 & 0.97 \end{bmatrix}, \quad \boldsymbol{\Sigma}_4 = \begin{bmatrix} 0.85 & 0.23 \\ 0.23 & 2.21 \end{bmatrix}$$

$$\lambda_1 = 5.01, \quad \lambda_2 = 5.05, \quad \lambda_3 = 5.08, \quad \lambda_4 = 4.83$$

$$\mathbf{A} = \begin{bmatrix} 0.22 & 0.36 & 0.21 & 0.21 \\ 0.10 & 0.74 & 0.09 & 0.07 \\ 0.07 & 0.17 & 0.70 & 0.06 \\ 0.32 & 0.48 & 0.17 & 0.03 \end{bmatrix}, \quad \boldsymbol{\pi}[0.25, 0.25, 0.25, 0.25]$$

Comme indiqué, la classe 4 n'est pas correctement estimée. Le vecteur $\boldsymbol{\pi}$ n'étant pas estimé, il reste fixé à $[1/K, \ldots, 1/K]$ (ce paramètre n'a de toute manière pas d'importance dans le cas en ligne)

Version incrémentale

On réalise dans les même conditions le partitionnement de ces valeurs avec l'algorithme *incrémental*. Le résultat est alors indiqué en figure III.5. Cette partition donne une F-mesure $65,3\%$. On constate toujours que les classes sont plutôt bien détectées à partir de t_{min} sauf la classe 4, on

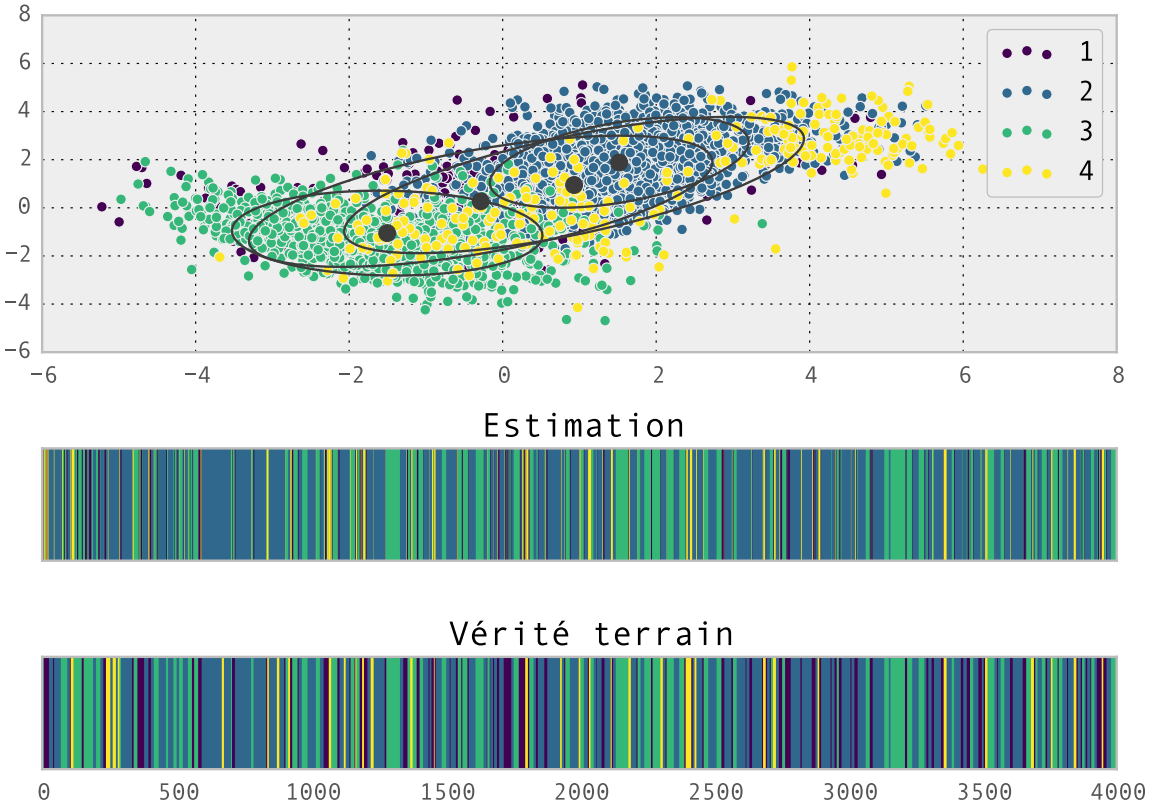


FIGURE III.5 – Partitionnement des données grâce à l'algorithme EM incrémental et comparaison avec la vérité-terrain

remarque tout de même une petite amélioration par rapport à l'algorithme précédent.

Les paramètres finaux sont :

$$\boldsymbol{\mu}_1 = \begin{bmatrix} -0.29 \\ 0.27 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 1.52 \\ 1.88 \end{bmatrix}, \quad \boldsymbol{\mu}_3 = \begin{bmatrix} -1.50 \\ -1.05 \end{bmatrix}, \quad \boldsymbol{\mu}_4 = \begin{bmatrix} 0.93 \\ 0.95 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 4.05 & 2.20 \\ 2.20 & 3.30 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 1.26 & 0.64 \\ 0.64 & 1.57 \end{bmatrix}, \quad \boldsymbol{\Sigma}_3 = \begin{bmatrix} 1.82 & -0.09 \\ -0.09 & 1.38 \end{bmatrix}, \quad \boldsymbol{\Sigma}_4 = \begin{bmatrix} 3.99 & 2.56 \\ 2.56 & 3.57 \end{bmatrix}$$

$$\lambda_1 = 4.67, \quad \lambda_2 = 4.96, \quad \lambda_3 = 4.95, \quad \lambda_4 = 4.80$$

$$\mathbf{A} = \begin{bmatrix} 0.04 & 0.25 & 0.38 & 0.33 \\ 0.08 & 0.52 & 0.14 & 0.26 \\ 0.49 & 0.22 & 0.23 & 0.06 \\ 0.39 & 0.45 & 0.11 & 0.05 \end{bmatrix}, \quad \boldsymbol{\pi} = [0.25, 0.25, 0.25, 0.25]$$

2. Segmentation d'un signal audio

1. Descripteurs audio

Comme la plupart des problèmes en traitement de signaux audio, les méthodes de partitionnement se basent sur une représentation temps-fréquence du signal à partitionner. Cette représentation permet de capturer les variations temporelles locales dans les composantes fréquentielles du signal.

Transformée de Fourier à Court Terme

La Transformée de Fourier à Court Terme (ou TFCT) est le descripteur audio le plus couramment utilisé en traitement du signal. Elle est basée sur la transformée de Fourier :

Définition III.1 Transformée de Fourier
Pour un signal audio $x[t] \in \mathbb{R}, t \in \mathbb{Z}$ la transformée de Fourier à temps discret est définie par :

$$\hat{x}[t] = \sum_{-\infty}^{+\infty} x[t] \mathrm{e}^{-i\omega t}$$

La transformée de Fourier seule ne permet cependant pas d'observer les variations fréquentielles locales. Pour ce faire, on utilisera donc la TFCT définie pour une fenêtre h.

Définition III.2 Transformée de Fourier à Court Terme
Soit $h : \mathbb{Z} \to \mathbb{R}$ une fonction de fenêtrage à support fini, la TFCT d'un signal $x[t] \in \mathbb{R}$ est

$$\hat{x}[t, \omega] = \sum_{u=-\infty}^{u=+\infty} x[u] h[u - t] \mathrm{e}^{-iwu}$$

Mel-Frequency Cepstrum Coefficients

Les *Mel-Frequency Cepstrum Coefficients* (MFCC) constituent une autre représentation possible d'un signal sur une échelle temps-fréquence (Zheng et al. 2001). Cette transformée se base sur une représentation en sous-bandes de fréquences également réparties sur l'échelle de Mel, définie par $M(f) = 1125 \ln(1 + \frac{f}{700})$. Ce comportement vise à reproduire le fonctionnement de l'audition humaine, séparant le son en sous-bandes critiques. Ceci fait des MFCC un descripteur souvent utilisé en reconnaissance de parole (Molla et al. 2004).

Les MFCC sont calculés selon le processus suivant :

1. Séparation de signal d'entrée en trames fenêtrées par un fonction h.
2. Calcul de l'estimée de la densité spectrale de puissance pour chaque trame par périodogramme : $x \to |\hat{x}|^2$.
3. Application d'un banc de filtre de Mel, et somme de l'énergie pour chaque filtre.
4. Prendre le logarithme de chaque énergie du banc de filtre.
5. Effectuer la Transformée en cosinus discret (DCT) de chaque logarithme.

2. Partitionnement

Nous présentons cette fois-ci un cas pratique. Nous montrons ici les résultats des différents algorithmes de partitionnement appliqués à la segmentation d'un signal audio, afin de donner un aperçu global des performances de chaque méthode. Nous utilisons pour cela un fichier audio tiré du jeu de fichiers utilisé pour l'évaluation (celui-ci sera présenté en détail en section III-3.1) Le fichier en question est une scène contenant des éléments de batterie (grosse caisse, caisse claire et charleston), et un fond sonore musical. L'idée est alors d'utiliser un algorithme de partitionnement pour détecter quel élément est joué à quel instant, et proposer une segmentation du signal de telle sorte que chaque segment corresponde à un unique élément, ou au fond sonore. La figure III.6 est une représentation du spectrogramme du ficher ainsi qu'une représentation des différents événements sonores qui le composent. Le charleston étant une cymbale, le son qu'il produit est assimilable à du bruit blanc, et est donc repérable sur le spectrogramme par les traits verticaux de forte intensité.

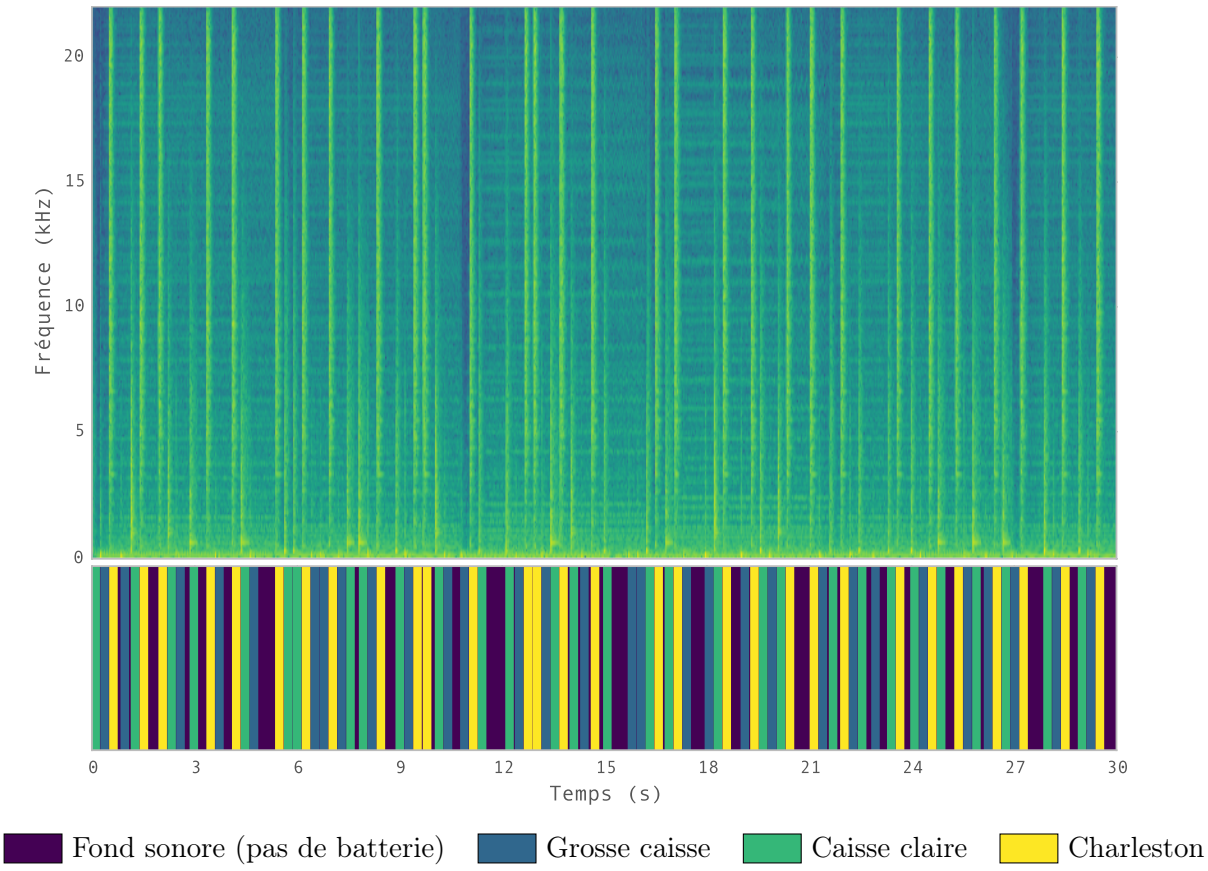


FIGURE III.6 – Spectrogramme et vérité-terrain de la scène à segmenter

La figure du haut représente un spectrogramme sur le plan temps fréquence, l'intensité de couleur est définie sur une échelle logarithmique. La figure du bas représente la segmentation du ficher, telle que définie par la vérité-terrain. Chaque bloc de couleur correspond à un événement associé (son produit par un composant de batterie ou fond sonore).

La caisse claire est repérable par un même trait vertical, atténué dans les hauts aigus, avec un pic d'intensité à quelques centaines de hertz. La grosse caisse est visible dans les graves fréquences, avec un court trait dû à l'attaque du son. Enfin le fond sonore, présent sur l'ensemble de l'extrait est responsable des traits horizontaux d'intensité moyenne. Il consiste en effet en des accords joués au synthétiseur, et produit donc des fréquences harmoniques.

Pour la segmentation, on utilisera le modèle HSMM, avec comme descripteur le spectrogramme utilisant des fenêtres de 2048 échantillons et un recouvrement de 75%. Le modèle d'émission est multinomial (donc divergence de Kullblack-Leibler) et le modèle de durée suit une loi de Poisson. On initialise les paramètres comme pour la section III-1.

3. Segmentation hors ligne

On utilise tout d'abord l'algorithme classique hors ligne, censé avoir les meilleures performances. La figure III.7 montre les résultats. Le résultat est globalement correct, le charleston est l'élément

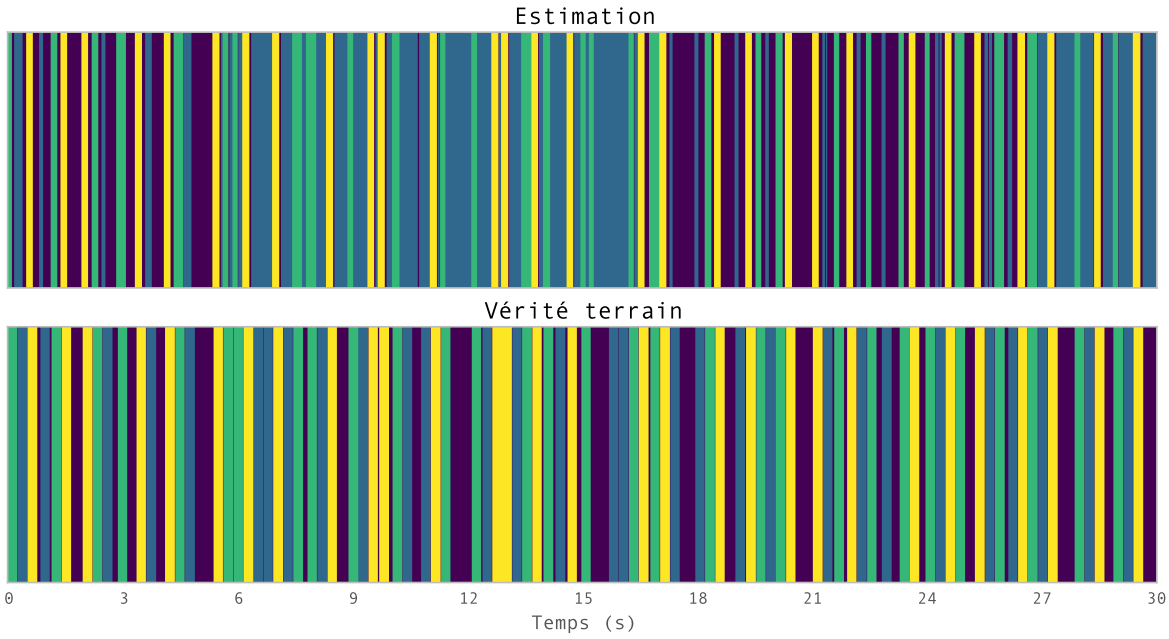


FIGURE III.7 – Segmentation hors-ligne du fichier audio

L'image du bas représente la segmentation du fichier audio selon la vérité terrain, l'image du haut la segmentation estimée par l'algorithme d'apprentissage HSMM, hors-ligne. Le code couleur utilisé est le même que pour la figure III.6.

le plus facilement détecté, par sa présence dans l'ensemble du spectre (comme nous l'avons vu avec la figure III.6). La caisse claire est, elle aussi, bien détectée. En revanche pour la grosse caisse, si les débuts des segments sont bien détectés, ils sont bien plus longs ou bien plus courts que sur la vérité-terrain. L'estimation indique donc sa présence là où il n'y a normalement plus que le fond sonore, et inversement. L'algorithme est donc très efficace si on cherche à détecter seulement le début des événements, ce qui peut être le cas pour des sons percussifs, comme ici.

Remarque On pourrait se dire que l'utilisation de MFCC comme descripteur est plus intéressante, puisque leur calcul se base sur une échelle de Mel (voir section III-2.1). S'ils permettent de mieux détecter la grosse caisse, les résultats que nous avons avec ce descripteur sont néanmoins globalement moins bons.

4. Segmentation en ligne

On effectue les mêmes opérations, mais cette fois-ci avec les deux algorithmes en ligne pour HSMM. Les centroïdes de chaque classe sont initialisés avec les 4 premiers vecteurs du spectre. La figure III.8 présente les résultats obtenus par les versions *en ligne* et *incrémentale*, ainsi qu'une analyse de ces résultats.

(a) Les résultats sont plutôt corrects : encore une fois le charleston est très bien détecté, la caisse claire aussi. On remarque aussi que le fond sonore n'est presque pas présent dans la segmentation estimée, il est presque toujours confondu avec la grosse caisse. Le début de la segmentation n'est pas correct du tout : cela correspond au temps t_{min} avant lequel on ne met pas à jour les paramètres.

b Les résultats sont plutôt similaires. En effet les deux algorithmes ont été initialisés de la même manière. On constate donc qu'il fonctionne de façon presque similaire. Les résultats sont tout de même légèrement différents pour la détection du fond sonore.

3. Évaluation des algorithmes

Nous présentons ici une évaluation des deux algorithmes, *en ligne* et *incrémental*, comparée à l'évaluation d'algorithmes de l'état de l'art. Tout d'abord l'ensemble de données utilisées pour l'évaluation, ainsi que les précédents résultats sont introduits, nous décrirons ensuite le protocole utilisé et enfin nous présenterons les résultats.

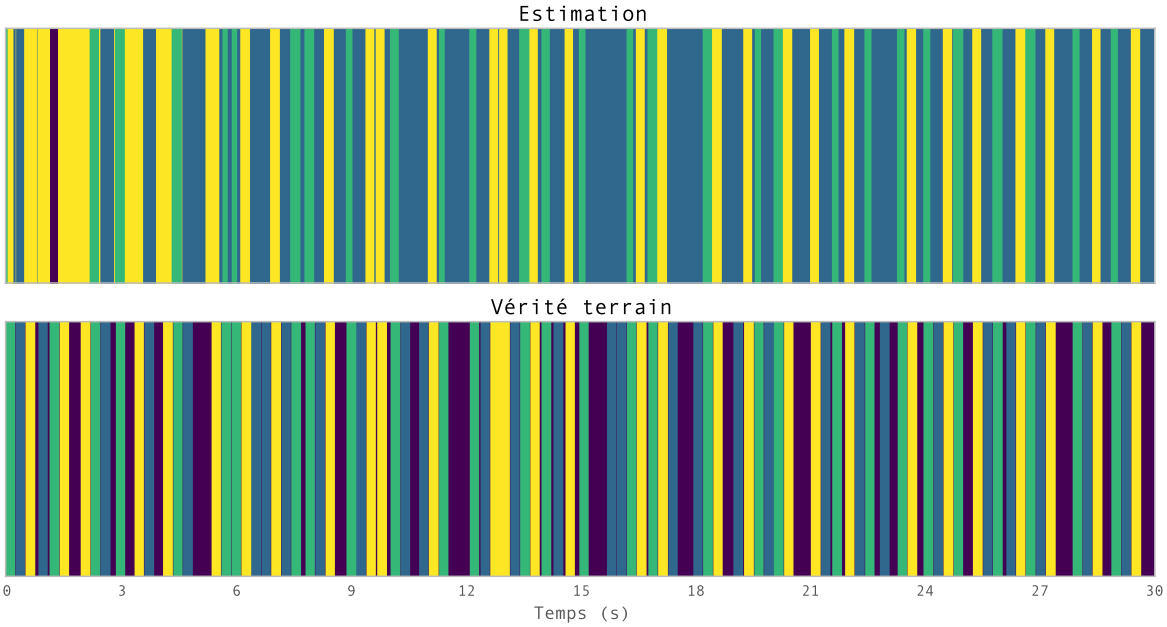
1. Jeu de données

Nous utilisons un jeu de scènes de batterie acoustique simulées pour mesurer les performances des algorithmes. Une scène acoustique est un fichier audio construit à partir d'enregistrements isolés de plusieurs « événements », et d'un fond sonore. Le jeu de scènes utilisé est celui proposé par ROSSIGNOL et al. (2015) pour l'évaluation de leur algorithme de transcription supervisé. Deux corpus de batterie distincts,tirés de la base de données ENST-*drums*[3], sont utilisés ici pour simuler les scènes. Chaque corpus consiste en plusieurs sons pour chaque composant d'une batterie, à savoir, la grosse caisse (*bass drum*, *bd*), la caisse claire (*snare drum*, *sd*) et le charleston (*hit-hat*, *hh*), qui sont les principaux éléments à détecter pour la transcription de batterie (PAULUS et al. 2005). Chaque son de batterie est coupé en utilisant une fenêtre de 250ms, avec un fondu de 10ms. On considère de plus deux fonds sonores différents, nommés bg_{easy} et bg_{hard}, consistant en une progression d'accords de nappes de synthétiseurs, le deuxième étant plus complexe.

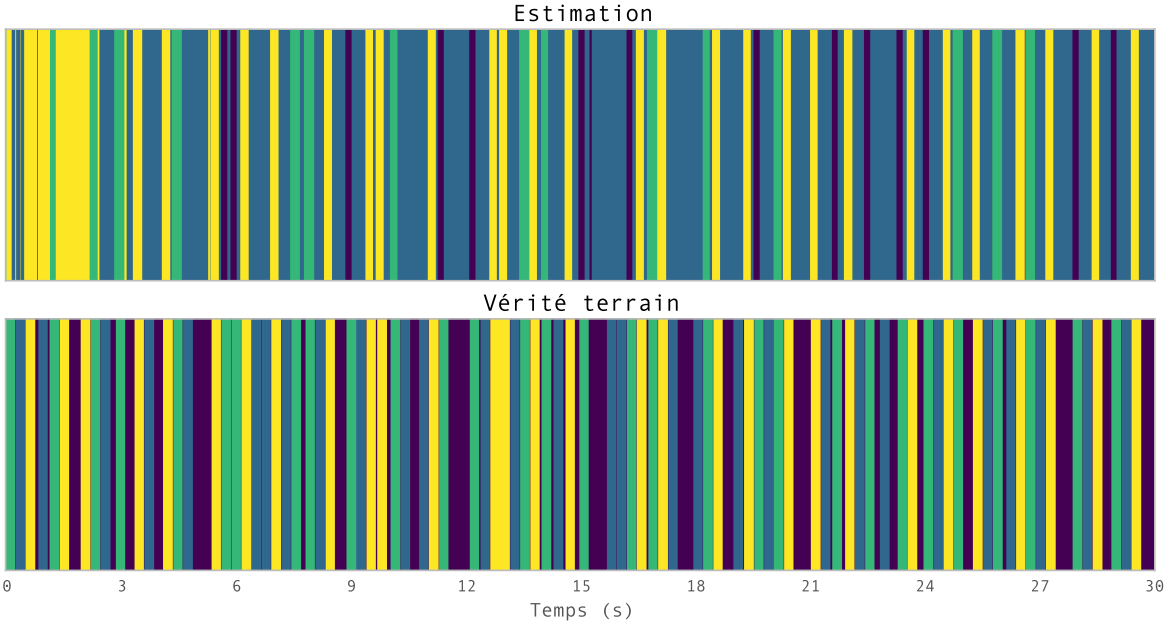
Les différentes scènes sont alors construites de la manière suivante : pour chaque classe d'événement (bd, sd, hh), on tire selon une loi normale de moyenne $\mu = 1$s et de déviation standard $\sigma = 0.25$s les durées entre chaque début d'événement, et pour chaque événement, on choisit le fichier audio utilisé en tirant un entier pseudo-aléatoire selon une loi uniforme. Deux jeux de données différents sont construits : un en utilisant le premier corpus de batterie et le fond bg_{easy}, utilisé par ROSSIGNOL et al. (2015) comme jeu de scènes d'entraînement pour leur algorithme supervisé, et un utilisant le second corpus et bg_{hard} comme jeu de scène de test. On utilise aussi pour chaque jeu de données 7 EBR[4] différents : $-24\,\text{dB}, -12\,\text{dB}, -6\,\text{dB}, 0\,\text{dB}, 6\,\text{dB}, 12\,\text{dB}, 24\,\text{dB}$.

Tous les fichiers construits sont encodés au format WAV en mono et à la fréquence d'échantillonnage $44,1$kHz.

3. http://www.tsi.telecom-paristech.fr/aao/2009/11/25/enst-drums-pistes-de-batterie-annotees
4. *Event to Background power Ratio*, le rapport de puissance entre les événements et le fond sonore



(a) Résultat avec la version *en ligne*



(b) Résultat avec la version *incrémentale*

FIGURE III.8 – Segmentation hors-ligne du fichier audio

2. Métriques

Nos algorithmes de partitionnement vont alors nous permettre de détecter les événements. On considère qu'un événement est détecté lorsqu'une série de trames contigües est classifiées uniformément (*e.g.* l'événement « charleston » est détecté sur la figure III.6 lorsqu'une tranche jaune apparaît dans l'estimation).

L'évaluation des algorithmes s'effectue en calculant la précision, le rappel et la F-mesure du partitionnement, définis classe à classe (Giannoulis, Benetos et al. 2013 ; Giannoulis, Stowell et al. 2013). On rappelle tout d'abord la définition de ces trois métriques dans le cas général. En recherche d'information, lorsqu'on cherche à identifier des éléments d'un ensemble possédant une propriété, on définit la précision comme le rapport des éléments identifiés qui sont pertinents, et le rappel comme le rapport des éléments pertinents qui sont correctement identifiés. En termes de partitionnement, si on définit r, e et c comme le nombre d'événements respectivement de la vérité-terrain, détectés et correctement détectés. Les métriques sont alors définies comme

$$P = \frac{c}{e},\ R = \frac{c}{r}$$

où P désigne la précision et et R le rappel. La F-mesure est alors définie comme la moyenne géométrique de ces deux valeurs

$$F = \frac{2PR}{P+R}$$

Dans notre cas, le métriques utilisées sont des métriques de frontière : on considère qu'un événement est correctement détecté si le le début estimé est dans une fenêtre de tolérance de 100ms autour du début de l'événement. La mesure classe à classe est alors effectuée en calculant P_k, R_k et F_k pour chaque classe. On peut alors définir la F-mesure classe à classe comme la moyenne des F_k pour l'ensemble des classes : $\frac{1}{K}\sum_k F_k$.

Alternativement on peut aussi définir ces mesures de manière paire à paire (Levy et al. 2008). Si on définit par P_E l'ensemble des paires formées de deux trames assignées à la même classe lors de l'estimation, et P_V les paires pour la vérité-terrain, on a alors

$$P_{paire} = \frac{|P_E \cap P_V|}{|P_E|},\ R_{paire} = \frac{|P_E \cap P_V|}{|P_V|}$$

F_{paire} est alors défini de la même manière.

3. Protocole

Tout d'abord, discutons des différents paramètres importants pour l'évaluation, leur notation et les valeurs possibles qu'ils peuvent prendre :

- Type de descripteur utilisé f_{eat} : TFCT ou MFCC (*c.f.* section III-2.1).
- Taille de la fenêtre pour le calcul des trames N_w : on choisit classiquement un nombre puissance de 2.
- Taux de recouvrement entre deux trames O_v.
- Initialisation des paramètres $\boldsymbol{\mu}$ pour l'émission, I_{nit} : tirés aléatoirement selon une loi normale ou initialisés à partir des K premières observations (on définit $\boldsymbol{\mu}_k = \mathbf{x}_k$).
- Pas d'incrémentation γ_t.
- Loi de probabilité pour les durées de séquence du modèle HSMM, D_{ur}. Cette loi doit être discrète et paramétrique de famille exponentielle afin de pouvoir mettre à jour les paramètres comme en (I.35). Deux distributions sont utilisées, la loi de Poisson et la loi Négative Binomiale (cette dernière possède deux paramètres r et p, permettant d'affiner la position et la taille du pic dans la courbe de densité de probabilité ; cependant on ne peut mettre à jour que le paramètre p, *c.f.* section I-6.2).

On modélise les observations par la loi multinomiale, ce qui revient à utiliser la divergence de Kullblack-Leibler comme nous l'avons précisé dans l'exemple I.4 (p.11). D'après la définition de cette distribution, il convient alors de normaliser chaque trame pour obtenir $\sum_i (x_t)_i = N \ \forall t$, où N est un paramètre que nous devons fixer.

Les algorithmes donnent en sortie un vecteur de même taille que le nombre de trames du descripteur. À chaque trame est associé un nombre de 1 à $K = 4$, correspondant à la classe estimée de la trame. Bien sûr ces numéros n'ont aucune signification par rapport aux types d'éléments à détecter. On utilise alors un l'algorithme Hongrois (Kuhn 1955). Ceci afin de trouver l'assignation optimale entre les numéros, correspondant aux classes identifiées, et l'événements de la vérité-terrain.

On va alors lancer les algorithmes, en faisant varier certains paramètres, et calculer la f-mesure classe à classe pour chaque scène. En faisant la moyenne de cette mesure pour chaque type d'ebr, on pourra comparer les résultats avec la courbe de référence publiée dans l'article original (Rossignol et al. 2015). On pourra aussi comparer les deux algorithmes, que ce soit en termes de performance de partitionnement, mais aussi en termes de rapidité.

4. Analyse des paramètres

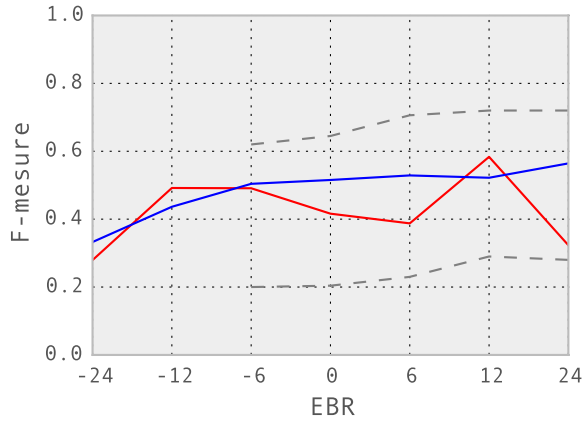
Nous avons tout d'abord voulu analyser l'importance de paramètres au moyen de la méthode d'analyse de variance anova (Chen 1988). En faisant varier les valeurs différents paramètres, cette méthode nous permet de déterminer si chaque paramètre à une influence sur la performance de l'algorithme (en utilisant les métriques présentées). Le jeu de données utilisé pour cette analyse est un jeu de scène acoustiques plus complexe que le jeu de batterie : ce sont des scènes simulées de la même manière utilisant des événements non musicaux (porte qui claque, clavier d'ordinateur, éternuement,...) (Lagrange et al. 2015). Les résultats n'ayant pas permis d'écarter des paramètres de la grille de recherche, nous ne les détaillerons pas ici. Il semble que tous les paramètres ont une importance, hormis l'initialisation des centroïdes $\boldsymbol{\mu}$.

4. Résultats

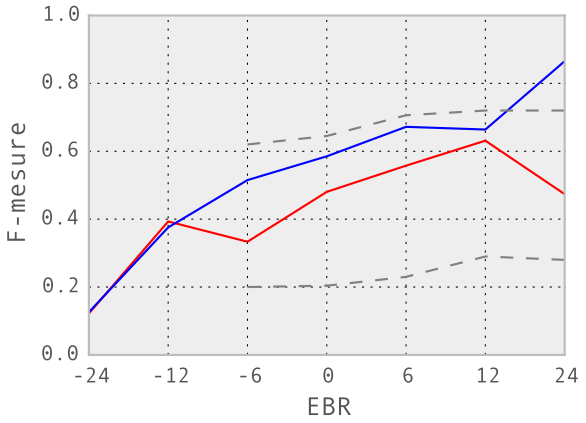
1. Résultats préliminaires

L'algorithme incrémental a d'abord été lancé avec sur l'ensemble du jeu de batterie, en faisant varier plusieurs paramètres. On fixe $K = 4$ et l'initialisation avec K premières observations. Les résultats sont séparés en deux catégories : ceux pour les scènes appartenant au jeu d'entraînement et ceux pour les scènes appartenant au jeu de test. La figure III.9 représentent certains résultats sous forme de courbes de la f-mesure en fonction de l'ebr. On affiche aussi les courbes de référence (Rossignol et al. 2015).

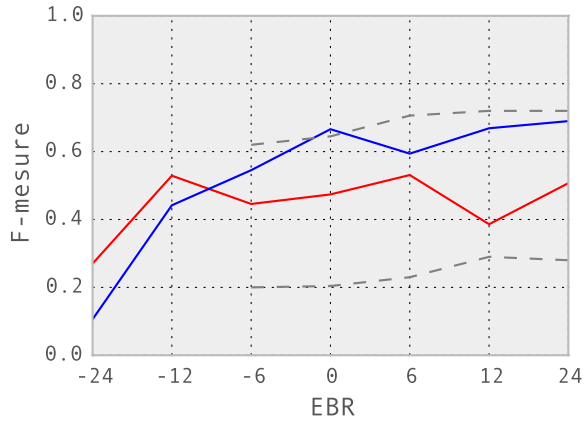
Plus l'ebr est faible, plus les événements sont difficiles à détecter (le fond sonore prend le dessus). Les courbes ont alors toutes une allure croissante. On considère les paramètres donnant les meilleurs résultats : on constate que les résultats sont bien meilleurs que ceux de l'article original. Les résultats sont légèrement en dessous, voire parfois au-dessus, de la courbe de référence pour le jeu d'entraînement. On précise que les algorithmes utilisés dans l'article original sont supervisés, et qu'ils sont entraînés avec ce jeu, d'où les bonnes performances. En revanche dans le cas du jeu de test, les performances sont bien moins bonnes pour la référence.



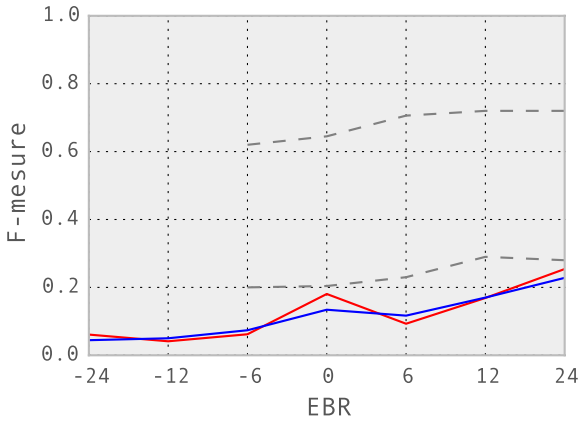
(a) N_w=512, O_v=50%, f_{eat}=spectr, D_{ur}=$\mathcal{N}(35, 0.5)$



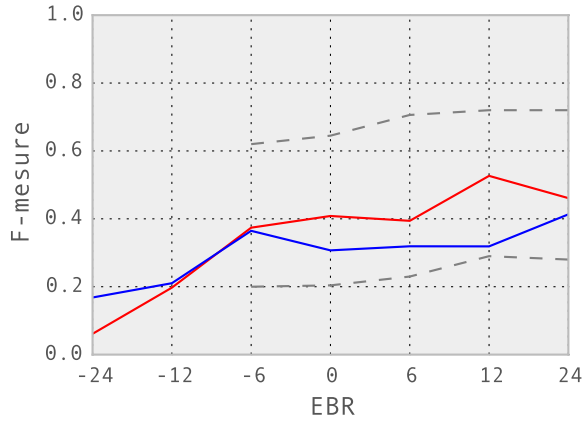
(b) N_w=2048, O_v=75%, f_{eat}=spectr, D_{ur}=$\mathcal{N}(4, 11)$



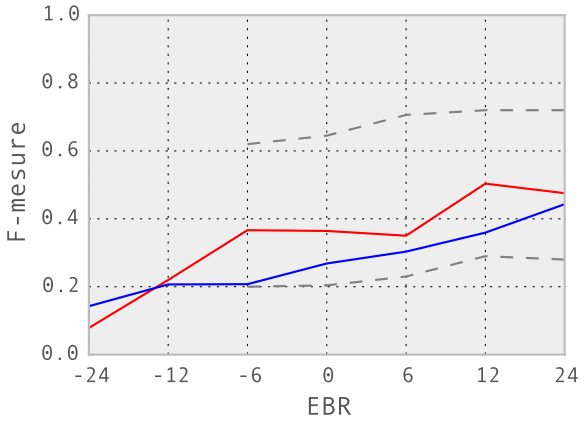
(c) N_w=2048, O_v=50%, f_{eat}=spectr, D_{ur}=$\mathcal{N}(35, 0.5)$



(d) N_w=8192, O_v=50%, f_{eat}=spectr, D_{ur}=$\mathcal{N}(35, 0.5)$



(e) N_w=2048, O_v=75%, f_{eat}=MFCC, D_{ur}=$\mathcal{P}(7.62)$



(f) N_w=2048, O_v=75%, f_{eat}=MFCC, D_{ur}=$\mathcal{P}(11.52)$

FIGURE III.9 – Performances de l'algorithme incrémental en fonction de l'EBR

Jeu de données 1
Jeu de données 2
Référence (jeu de données 1 - haut, et 2 -bas)

2. Distribution de durée et descripteurs

À la vue des résultats que nous avons obtenus, nous avons choisi de fixer les paramètres $N_w = 2048$ et $O_v = 75\%$ pour les algorithmes (pour une fréquence d'échantillonnage de 44, 1kHz, cela correspond à des trames d'environ 46ms et un décalage de $12ms$ entre chaque trame).

On lance alors l'algorithme en faisant varier les distributions de durée sur une plage de valeurs intéressantes, pour les deux descripteurs et les deux algorithmes. Les événements à détecter étant d'une durée de $d = 0, 25$s, nous avons décidé de choisir les densités de probabilité de durée ayant leur espérance en d, mais aussi celles ayant leur espérance en $d/2$ et $d/3$. En effet le modèle HSMM que nous utilisons autorise une transition vers le même état à la fin d'une séquence. On cherche donc à savoir s'il n'est pas préférable de fixer des modèles de durée plus court, en laissant le modèle sélectionner le même état plusieurs fois de suite. On choisit aussi une distribution d'espérance très courte.

La figure A.2 et la figure A.1, affichées en annexe A, montrent les densités des distributions choisies pour les mesures, avec les paramètres associés. La distribution binomiale négative ayant deux paramètres, on peut aussi régler les paramètres pour avoir une variance plus ou moins importante (voir tableau I.3). Pour la distribution de Poisson la variance est fixée et vaut λ. La durée d'une séquence est définie par le nombre de trames qui la composent, pour la convertir en secondes, on utilise les valeurs de fenêtrage, de recouvrement et de fréquence d'échantillonnage ($t = d * N_w * O_v/F_e$, où t est la durée en secondes, d est la durée en nombre de trame et F_e est la fréquence d'échantillonnage).

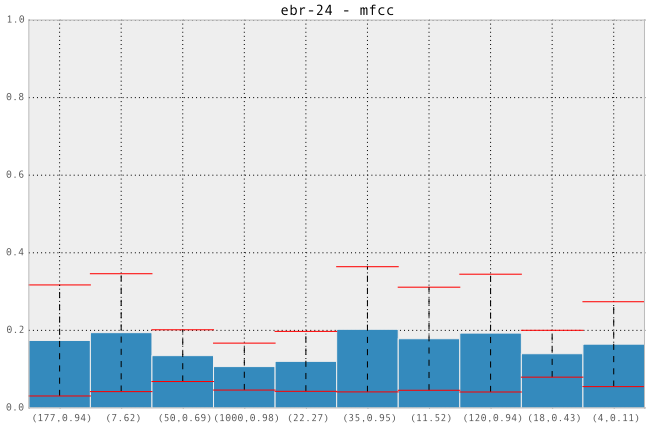
On présente tout d'abord les résultats pour les spectrogrammes. La figure III.10 et la figure III.11 donnent les résultats utilisant comme descripteur respectivement les MFCC et le spectrogramme. Ces résultats sont affichés sous forme d'histogramme, chaque barre correspondant à la moyenne de la F-mesure pour une distribution de durée donnée. On présente les résultats pour chaque EBR. Les déviations standards sont aussi représenté par les traits rouges. Un détails de ces résultats est présenté en annexe B avec le tableau B.1, les résultats de référence y sont aussi présents. On effectue ensuite les même tests en utilisant l'algorithme *en ligne*, avec le spectrogramme comme descripteur (figure III.12).

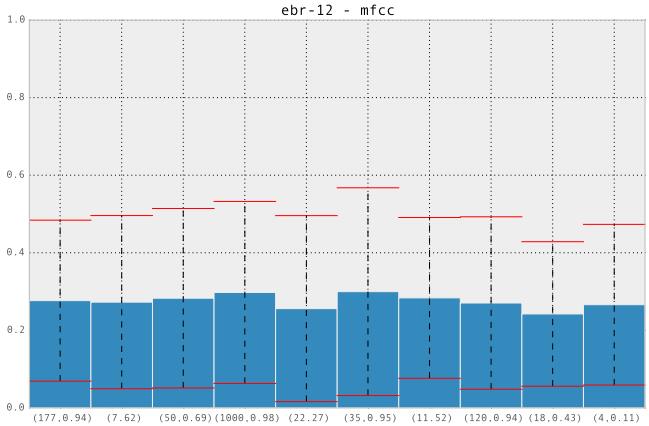
3. Analyse des résultats

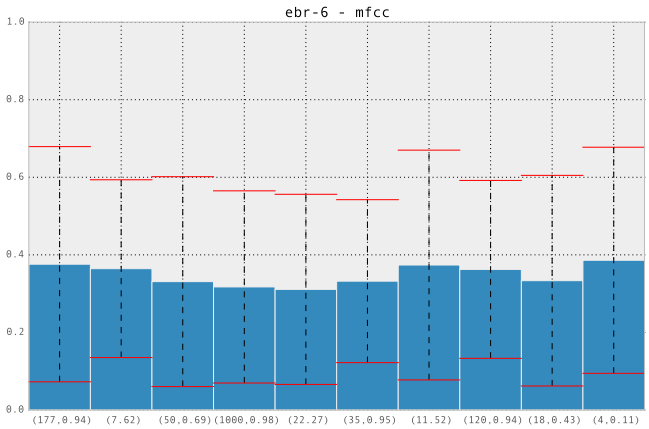
MFCC Les résultats sont cohérents puisque l'algorithme et plus performant pour les EBR élevés. Ils sont aussi meilleurs que la référence pour le jeu de test, même s'ils restent moins bon que la référence d'entraînement. On voit clairement que les modèles de durée ayant une espérance courte donnent en général des résultats meilleurs : le modèle $\mathcal{N}(38, 0.95)$ est parmi les plus performants. Comme on peut les voir sur la figure A.2, son espérance est très courte. Ceci pose la question de l'utilité de HSMM lorsqu'on utilise ce descripteur, en effet utiliser un modèle tel que celui-ci revient presque à estimer le modèle comme un HMM.

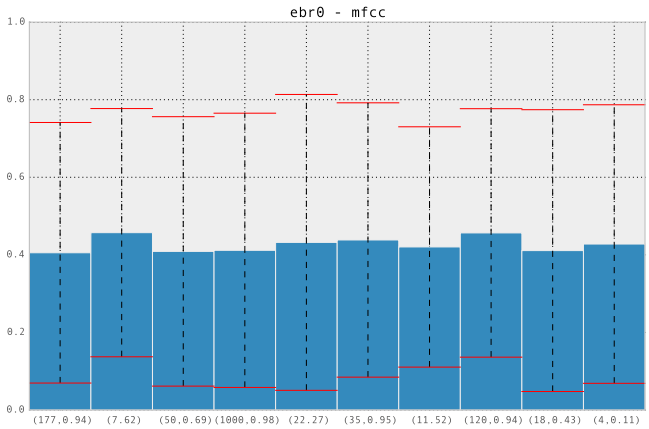
Spectrogramme Ici les résultats sont bien meilleurs que pour les MFCC. On a même des résultats plus performants que la référence appliquée au jeu de données d'entraînement, notre algorithme surpasse donc cette méthode supervisée. Le résultat dépend peu de la distribution de durée et de ses paramètres initiaux, ce qui peut vouloir dire que l'apprentissage des durées se fait correctement. Il faut rester néanmoins conscient que le modèle à apprendre ici est très simple puisque trois événements parmi quatre – les trois événements de batterie – ont des durées très similaires et fixes

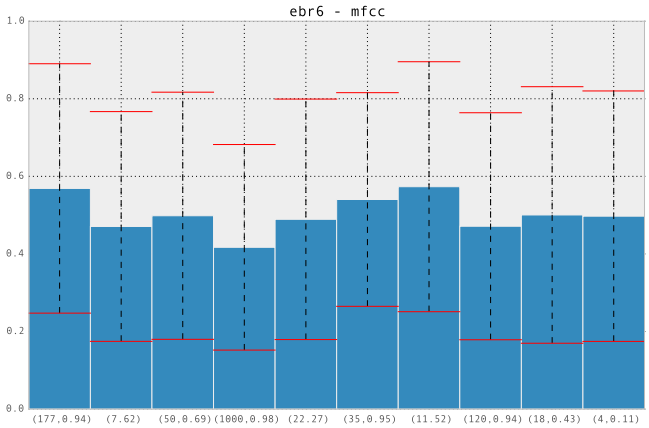
En ligne Les résultats sont sensiblement les mêmes que pour l'algorithme *incrémental*, avec le spectrogramme. Ces résultats ne sont pas trop dépendants de la distribution de durée non plus.

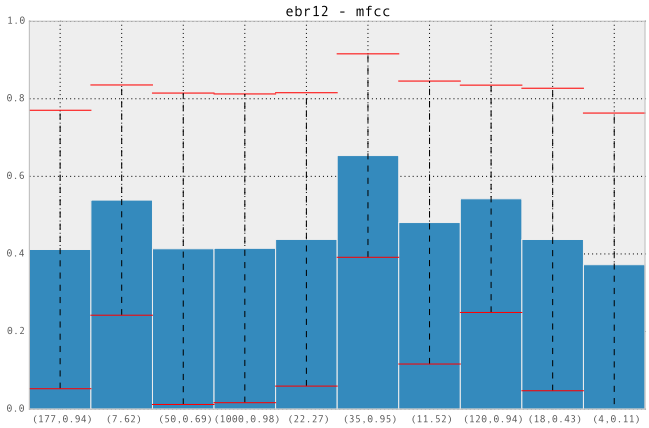












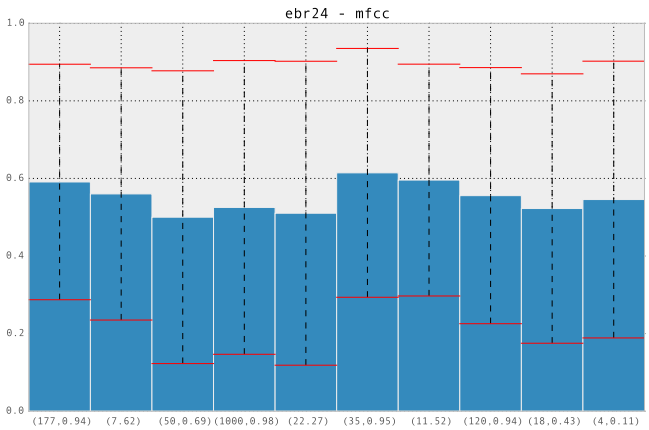
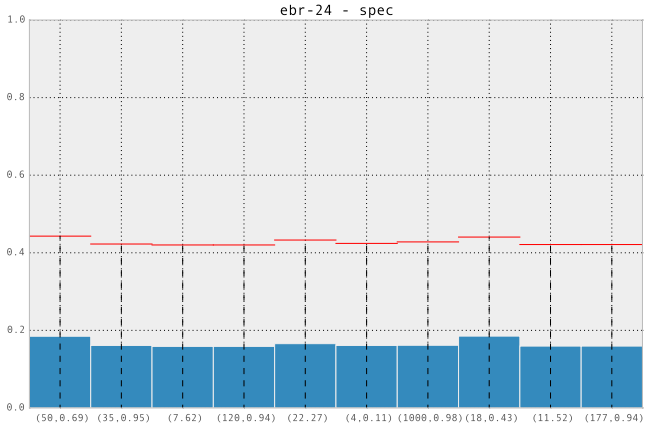
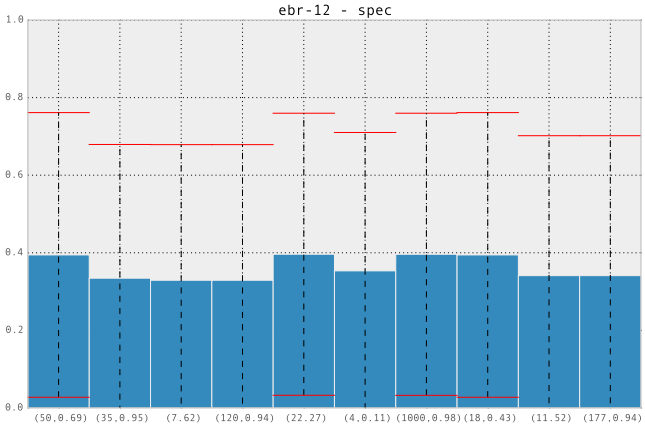
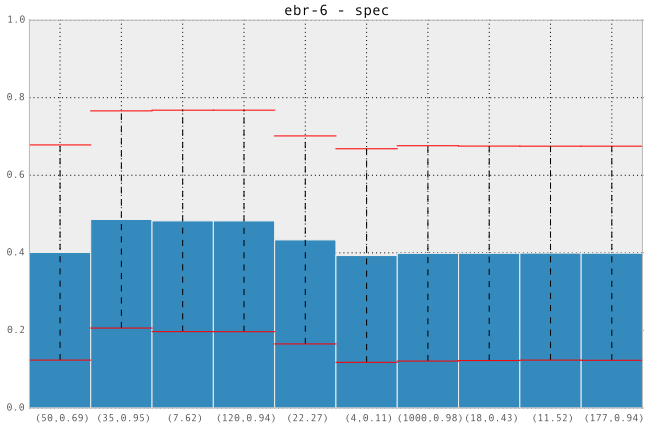
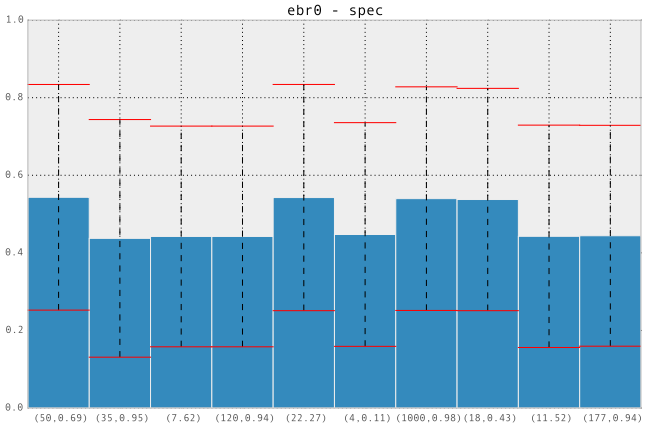


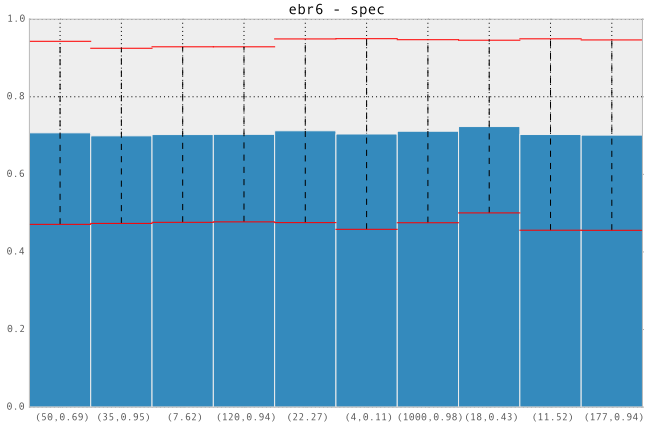
FIGURE III.10 – Moyenne et déviation standard par distribution et EBR pour les MFCC, algorithme *incrémental*

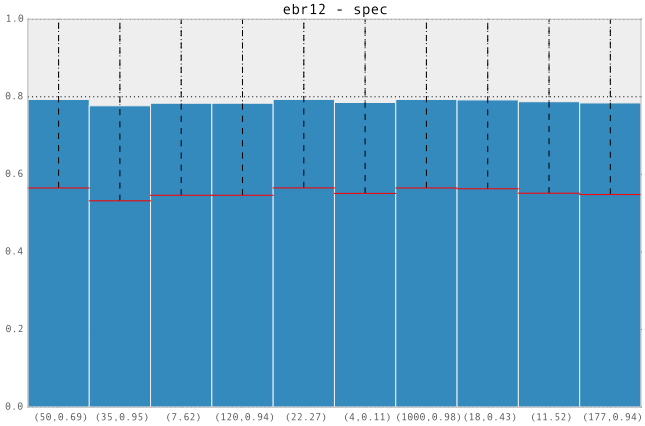












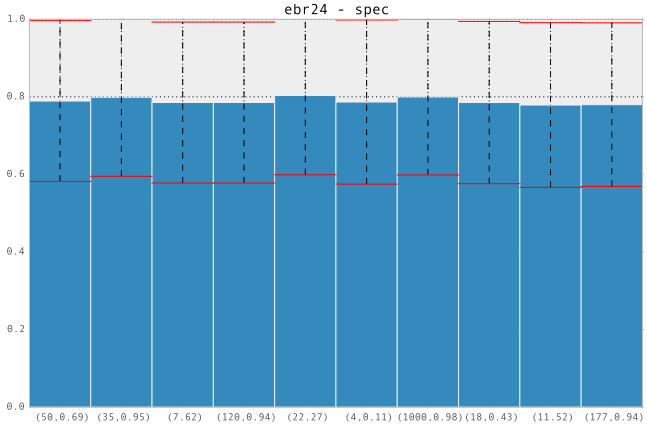


FIGURE III.11 – Moyenne et déviation standard par distribution et EBR pour le spectrogramme, algorithme *inc*
mental

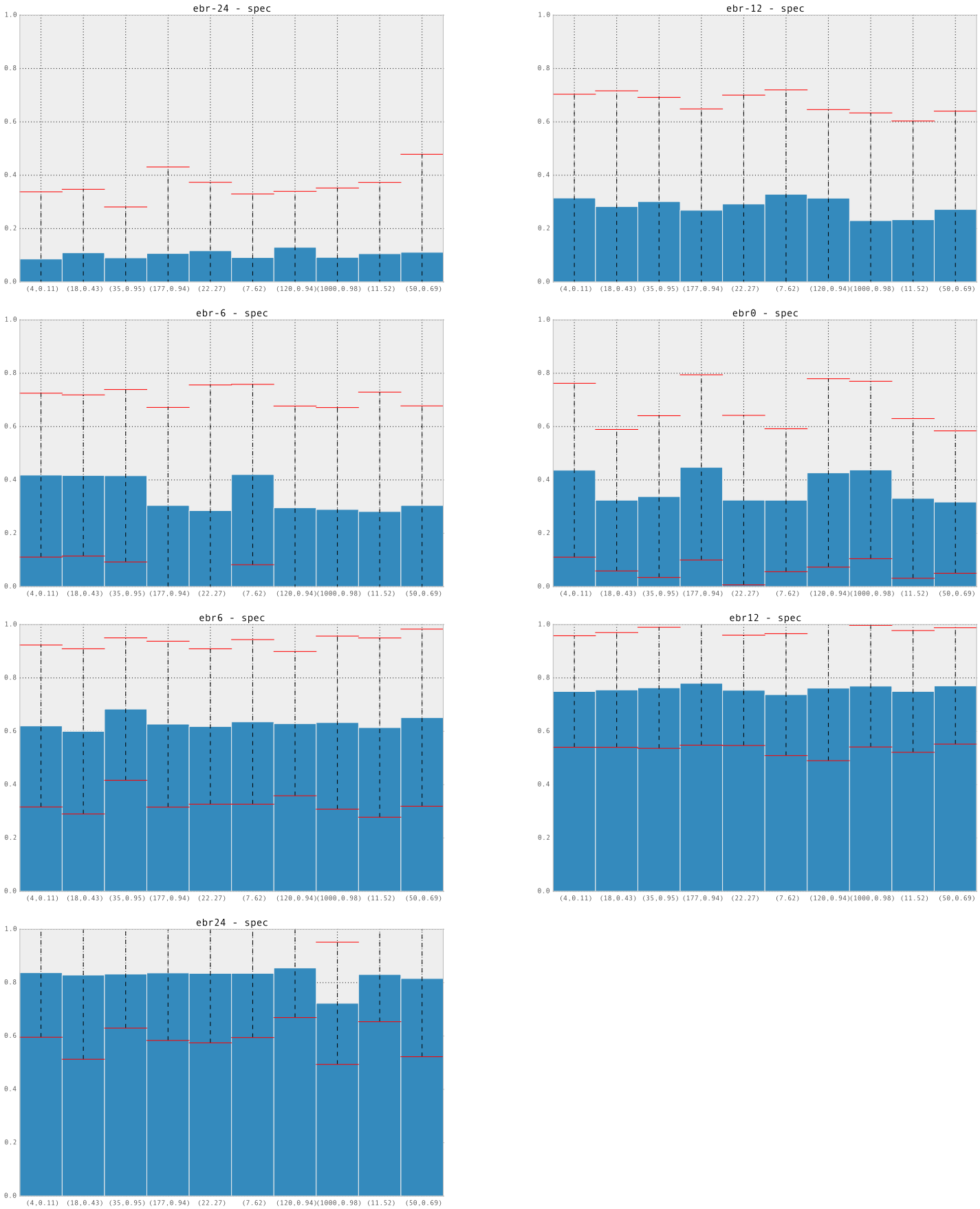


FIGURE III.12 – Moyenne et déviation standard par distribution et EBR pour le spectrogramme, algorithme *en ligne*

4. Performances en temps

On a vu que l'algorithme *incrémental* avait une complexité temporelle relativement moindre par rapport à l'algorithme *en ligne*. Pour vérifier si cette différence est vraiment marquante, nous avons exécuté les deux versions en faisant varier la dimension des observations et le nombre K de classes à détecter sur un ensemble de scènes de durée de 30 secondes. Ces deux valeurs, avec le nombre d'observations, entrent en compte dans le calcul de la complexité. Le tableau III.1 présente les résultats. Il est indéniable que l'algorithme *en ligne* a un temps d'exécution relativement long par rapport à la durée de la scène, et ce, même pour des petites valeurs. Pour K relativement élevé, il en devient même inutilisable et son temps d'exécution relève de l'ordre de l'heure pour à peine une quinzaine de classes à détecter. En revanche l'algorithme *incrémental* est lui très rapide. Il varie très peu en fonction de la dimension. Lorsque le nombre classe à détecter est inférieur à une vingtaine, on peut considérer qu'une application temps-réel est possible.

K	Incrémental (s)	Online (s)
4	2.03045	92.41348
8	4.90913	441.965520
12	8.9786	1267.60762
16	14.2315	...
20	20.2896	...
24	28.1308	...
28	37.392	...

(a) Nombre de classes à détecter variable

Dim	Incrémental (s)	Online (s)
128	4.76	38.25
256	4.62	68.00
384	4.88	99.73
512	5.10	141.62
640	5.24	160.23
748	5.95	193.51
896	5.52	314.73
1024	5.64	285.35

(b) Nombre de dimensions variable

TABLE III.1 – Temps d'exécution comparés

a Le nombre de dimensions des observations vaut 1024.

b Le nombre de classes à détecter vaut 6.

Conclusion

Ce stage a été l'occasion de travailler sur une nouvelle méthode de partitionnement en ligne, appliquée à la segmentation de signaux audio. De nombreux travaux ont été effectués au sein de l'équipe MuTant en ce qui concerne ce domaine, notamment pour les recherches orientées autour de l'outil de suivi de partition *Antescofo*. Après avoir cherché et mis en place des méthodes utilisant la géométrie de l'information (Cont et al. 2011 ; Dessein et al. 2013 ; Lostanlen 2013), une orientation basée sur des méthodes de partitionnement ont été proposées (Cuvillier et al. 2014). Les travaux initiés par Bietti et al. (2015) ont ouvert la voie pour de nouvelles méthodes de partitionnement, de manière incrémentale. L'objectif de ce stage était donc de mettre en pratique ces travaux théoriques, et d'évaluer ces algorithmes.

Nous avons tout d'abord défini au chapitre I les outils sur lesquels nous nous appuyons pour définir ces méthodes : les divergences de Bregman qui permettent de faire un parrallèle entre les mesures de similarité et la disitribution modélisant les données, et l'algorithme em, algorithme classique en apprentissage statistique, pour les modèles de mélange, hmm et hsmm. Dans le chapitre II nous avons présenté les version *en ligne* et *incrémentale* de l'algorithme em pour ces même modèles. La version *en ligne*, inspiré de Cappé (2011) et Cappé et Moulines (2009), assure la convergence de l'algorithme, au dépend de la complexité, d'ordre bien plus élevé que pour la version *incrémentale*. Enfin dans le chapitre III, nous avons présenté quelques exemples de résultats, ainsi que les mesures des performances des algorithmes implémentés. Après avoir introduit le protocole, d'évaluation et les métriques utilisées, nous avons présenté les performances des méthodes appliquées à a transcription de scènes de batterie simulées.

Les résultats que nous avons obtenus sont très encourageants. En effet ceux-ci sont bien meilleurs que ceux de la référence (Rossignol et al. 2015), quand bien même ces derniers sont obtenus avec un algorithme supervisé. La version *incrémentale* reste la plus intéressante du fait de sa rapidité d'exécution, étant donné l'objectif de pouvoir appliquer cette méthode en temps réel. La version *en ligne*, si elle présente tout de même des résultats tout à fait intéressants, est énormément handicapée par sa durée d'exécution. En effet les différences d'ordre de complexité que nous avons soulevées au chapitre II impliquent un temps d'exécution beaucoup plus important pour cette version. Il faut néanmoins garder en tête le fait que le jeu de scènes utilisé reste tout de même simple, et n'a pas posé trop de problèmes au niveau de la modélisation des durées. La transcription de scènes de batterie implique de détecter des événements de durées très proches, et constantes tout au long d'une scène. On pourrait pousser l'analyse avec des scènes acoustiques complexes, constituées d'événements de durées très largement différentes. Ceci permet tout de même de confirmer l'efficacité des algorithmes en ce qui concerne la détection d'événements sonores, et de la possibilité de les utiliser pour une tâche de segmentation audio.

On peut alors envisager des perspectives intéressantes pour l'algorithme hsmm de partitionnement *incrémental*. Sous réserve d'une confirmation de ses performances, il serait intéressant de pouvoir le mettre en application, en l'intégrant dans un outil tel que *Antescofo*. Cet outil est pour l'instant restreint à l'utilisation de *template* audio pour la détection des événements et implique une certaine stationnarité de ces événements. On pourrait utiliser cette nouvelle méthode afin de l'étendre à la détection d'événements sonores non stationnaires et non musicaux. Cette méthode pourrait aussi être envisagée pour être intégrée dans des environnements d'apprentissage statistique dédié à l'écoute artificielle, et temps réel. La version *en ligne* est pour l'instant inutilisable en temps-

réel, en raison de sa complexité d'ordre trop élevé. En revanche, au vu de ces performances comparées à l'autre version, on pourrait tout de même en tenir compte.

Bibliographie

Banerjee, Arindam, Xin Guo et Hui Wang (2004). « Optimal Bregman prediction and Jensen's equality ». In : *IEEE International Symposium on Information Theory*. Citeseer, p. 169–169.

Banerjee, Arindam, Srujana Merugu, Inderjit S. Dhillon et Joydeep Ghosh (2005). « Clustering with Bregman Divergences ». In : *J. Mach. Learn. Res.* 6, p. 1705–1749.

Bietti, Alberto (2014). « Online learning for audio clustering and segmentation ». other. ENS Cachan.

Bietti, Alberto, Francis Bach et Arshia Cont (2015). « An online EM algorithm in hidden (semi-) Markov models for audio segmentation and clustering ». In : *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Bregman, L. M. (1967). « The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming ». In : *USSR Computational Mathematics and Mathematical Physics* 7.3, p. 200–217.

Cappé, Olivier (2011). « Online EM Algorithm for Hidden Markov Models ». In : *Journal of Computational and Graphical Statistics* 20.3, p. 728–749.

Cappé, Olivier et Eric Moulines (2009). « On-line expectation–maximization algorithm for latent data models ». In : *Journal of the Royal Statistical Society : Series B (Statistical Methodology)* 71.3, p. 593–613.

Chen, Cheng-Liang (1988). « Analysis of variance ». In :

Cont, A., Shlomo Dubnov et G. Assayag (2011). « On the Information Geometry of Audio Streams With Applications to Similarity Computing ». In : *IEEE Transactions on Audio, Speech, and Language Processing* 19.4, p. 837–846.

Cuvillier, P. et A. Cont (2014). « Coherent time modeling of Semi-Markov models with application to real-time audio-to-score alignment ». In : *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. 2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP), p. 1–6.

Dellaert, Frank (2002). « The expectation maximization algorithm ». In :

Dempster, A. P., N. M. Laird et D. B. Rubin (1977). « Maximum Likelihood from Incomplete Data via the EM Algorithm ». In : *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1, p. 1–38.

Dessein, Arnaud et Arshia Cont (2013). « An Information-Geometric Approach to Real-Time Audio Segmentation ». In : *IEEE Signal Processing Letters* 20.4, p. 331–334.

Dhillon, Inderjit S., Subramanyam Mallela et Rahul Kumar (2003). « A divisive information theoretic feature clustering algorithm for text classification ». In : *The Journal of Machine Learning Research* 3, p. 1265–1287.

Giannoulis, Dimitrios, Emmanouil Benetos, Dan Stowell, Mathias Rossignol, Mathieu Lagrange et Mark D. Plumbley (2013). « Detection and classification of acoustic scenes and events : an IEEE AASP challenge ». In : *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*. IEEE, p. 1–4.

Giannoulis, Dimitrios, Dan Stowell, Emmanouil Benetos, Mathias Rossignol, Mathieu Lagrange et Mark D. Plumbley (2013). « A database and challenge for acoustic scene classification and event detection ». In : *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*. IEEE, p. 1–5.

Guédon, Yann (2003). « Estimating Hidden Semi-Markov Chains From Discrete Sequences ». In : *Journal of Computational and Graphical Statistics* 12.3, p. 604–639.

Kemp, Thomas, Michael Schmidt, Martin Westphal et Alex Waibel (2000). « Strategies for automatic segmentation of audio data ». In : *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. T. 3. IEEE, p. 1423–1426.

Kraskov, Alexander, Harald Stögbauer, Ralph G. Andrzejak et Peter Grassberger (2003). « Hierarchical Clustering Based on Mutual Information ». In : *arXiv :q-bio/0311039*. arXiv : q-bio/0311039.

Kuhn, H. W. (1955). « The Hungarian method for the assignment problem ». In : *Naval Research Logistics Quarterly* 2.1, p. 83–97.

Lagrange, Mathieu, Grégoire Lafay, Mathias Rossignol, Emmanouil Benetos et Axel Roebel (2015). « An evaluation framework for event detection using a morphological model of acoustic scenes ». In :

Lange, Kenneth (1995). « A Gradient Algorithm Locally Equivalent to the EM Algorithm ». In : *Journal of the Royal Statistical Society. Series B (Methodological)* 57.2, p. 425–437.

Levy, M. et M. Sandler (2008). « Structural Segmentation of Musical Audio by Constrained Clustering ». In : *IEEE Transactions on Audio, Speech, and Language Processing* 16.2, p. 318–326.

Lostanlen, Vincent (2013). « Découverte automatique de structures musicales en temps réel par la géométrie de l'information ». other. ATIAM, University of Paris 6 (UPMC).

MacQueen, J. (1967). « Some methods for classification and analysis of multivariate observations ». In : Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1 : Statistics. The Regents of the University of California.

Mitchell, Carl D. et Leah H. Jamieson (1993). « Modeling duration in a hidden Markov model with the exponential family ». In : *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*. T. 2. IEEE, p. 331–334.

Molla, Md.K.I. et K. Hirose (2004). « On the effectiveness of MFCCs and their statistical distribution properties in speaker identification ». In : *2004 IEEE Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2004. (VECIMS)*. 2004 IEEE Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2004. (VECIMS), p. 136–141.

Murphy, Kevin P. (2002). « Hidden semi-markov models (hsmms) ». In : *unpublished notes* 2.

Neal, Radford M. et Geoffrey E. Hinton (1998). « A View of the Em Algorithm that Justifies Incremental, Sparse, and other Variants ». In : *Learning in Graphical Models*. Sous la dir. de Michael I. Jordan. NATO ASI Series 89. Springer Netherlands, p. 355–368.

Nielsen, F. et R. Nock (2009). « Sided and Symmetrized Bregman Centroids ». In : *IEEE Transactions on Information Theory* 55.6, p. 2882–2904.

Paulus, Jouni et Tuomas Virtanen (2005). « Drum transcription with non-negative spectrogram factorisation ». In : *Signal Processing Conference, 2005 13th European*. IEEE, p. 1–4.

Rabiner, L. (1989). « A tutorial on hidden Markov models and selected applications in speech recognition ». In : *Proceedings of the IEEE* 77.2, p. 257–286.

Rabiner, L.R., C.H. Lee, B.H. Juang et J.G. Wilpon (1989). « HMM clustering for connected word recognition ». In : *, 1989 International Conference on Acoustics, Speech, and Signal Processing, 1989. ICASSP-89.* , 1989 International Conference on Acoustics, Speech, and Signal Processing, 1989. ICASSP-89, 405–408 vol.1.

Rockafellar, R. Tyrrell (1970). « Convex Analysis (Princeton Mathematical Series) ». In : *Princeton University Press* 46, p. 49.

Rossignol, Mathias, Mathieu Lagrange, Grégoire Lafay et Emmanouil Benetos (2015). « Alternate Level Clustering for Drum Transcription ». In :

Smyth, Padhraic (1997). « Clustering Sequences with Hidden Markov Models ». In : *Advances in Neural Information Processing Systems*. MIT Press, p. 648–654.

Titterington, D. M. (1984). « Recursive Parameter Estimation Using Incomplete Data ». In : *Journal of the Royal Statistical Society. Series B (Methodological)* 46.2, p. 257–267.

Tzanetakis, G. et P. Cook (1999). « Multifeature audio segmentation for browsing and annotation ». In : *1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, p. 103–106.

Yu, Shun-Zheng (2010). « Hidden semi-Markov models ». In : *Artificial Intelligence*. Special Review Issue 174.2, p. 215–243.

Zheng, Fang, Guoliang Zhang et Zhanjiang Song (2001). « Comparison of different implementations of MFCC ». In : *Journal of Computer Science and Technology* 16.6, p. 582–589.

Annexe A

Distributions de durées utilisées

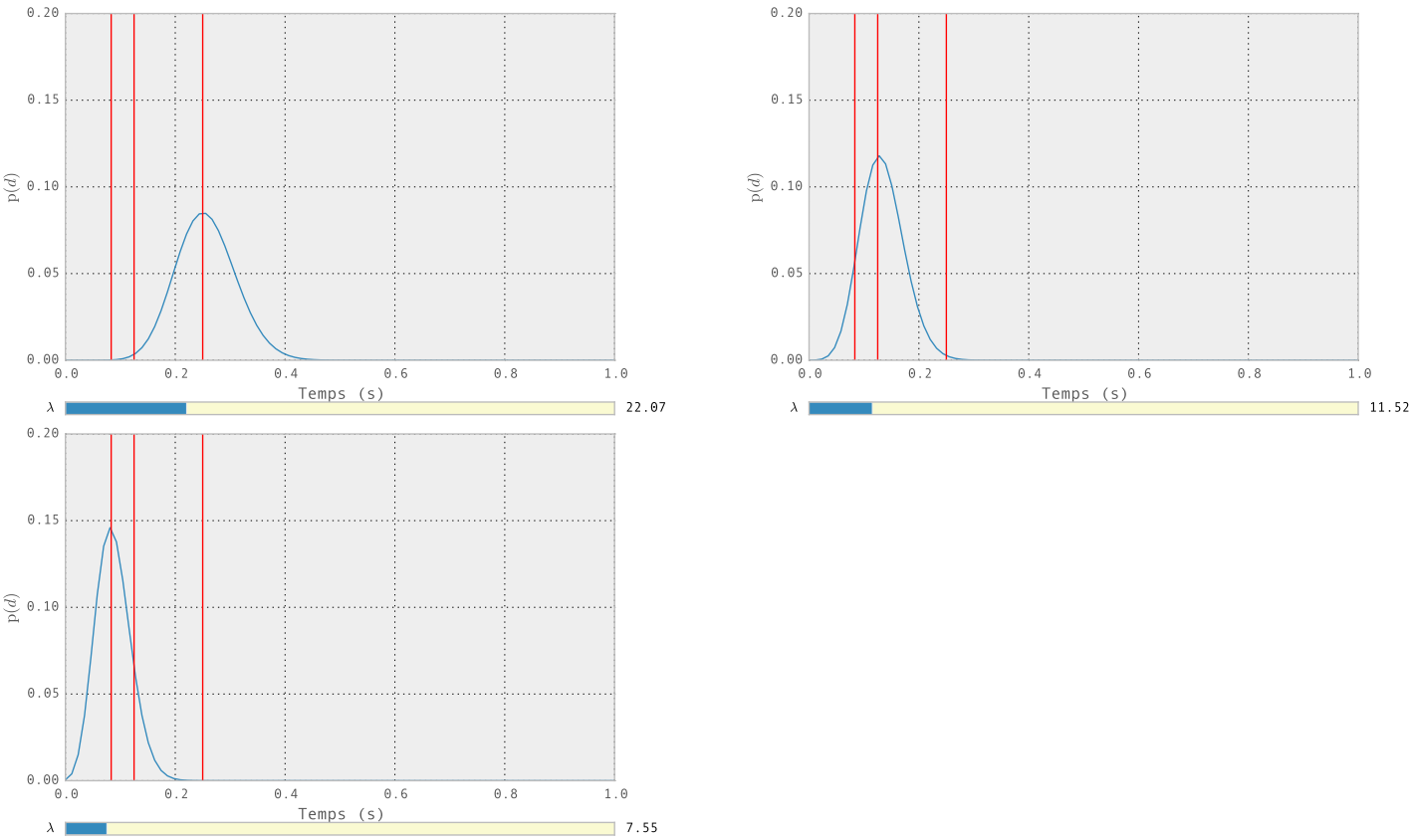


FIGURE A.1 – Visualisation des densité selon d pour la loi de Poisson

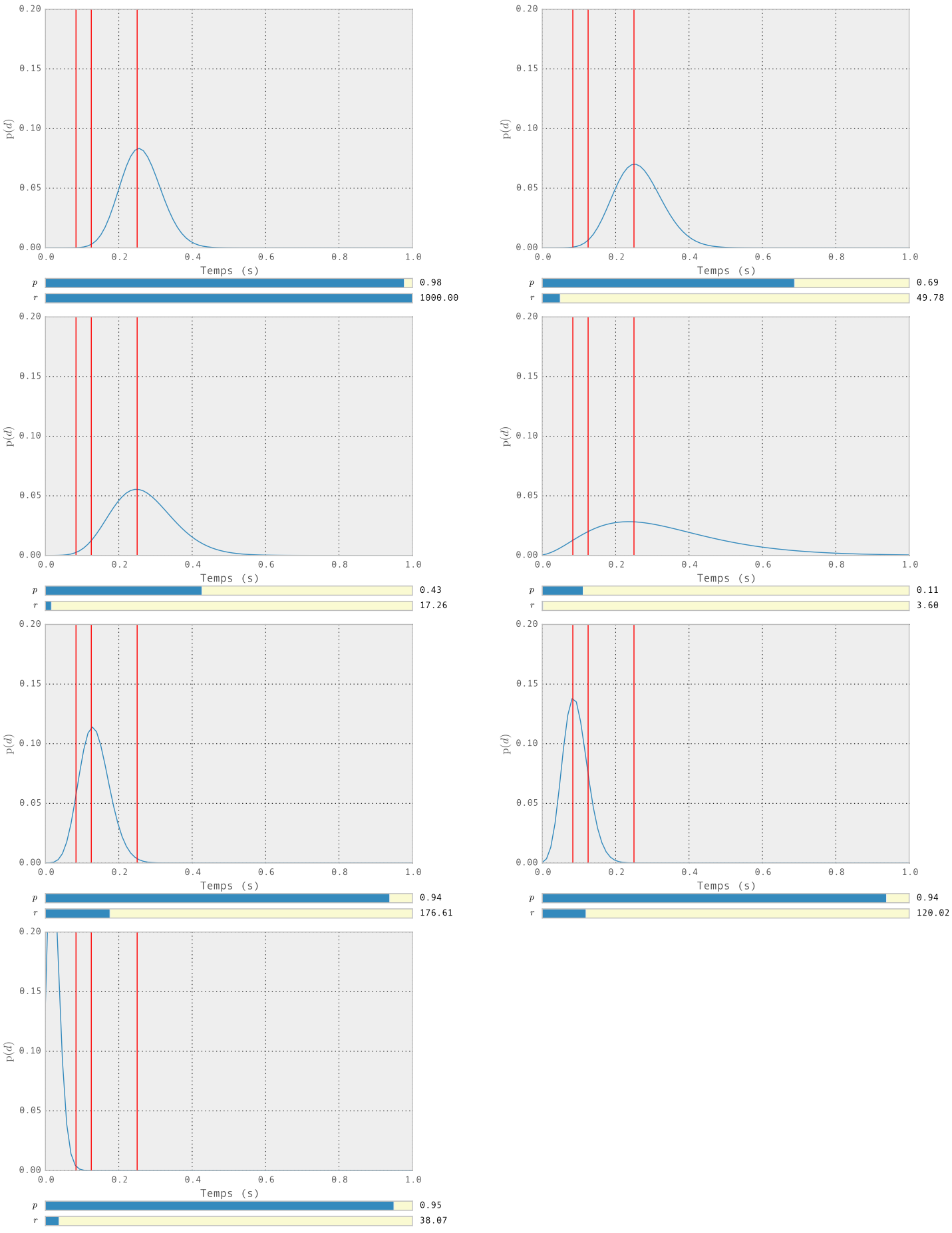


FIGURE A.2 – Visualisation des densité selon d pour la loi négative binomiale

Annexe B

Résultats expérimentaux : valeurs numériques

Distribution de durée	EBR-24	EBR-12	EBR-6	EBR0	EBR6	EBR12	EBR24
$\mathcal{N}(177, 0.94)$	17 ± 14	28 ± 21	38 ± 30	41 ± 34	57 ± 32	41 ± 36	59 ± 30
$\mathcal{P}(11.52)$	18 ± 13	28 ± 21	37 ± 30	42 ± 31	57 ± 32	48 ± 36	60 ± 30
$\mathcal{N}(4, 0.11)$	16 ± 11	27 ± 21	39 ± 29	43 ± 36	50 ± 32	37 ± 39	55 ± 36
$\mathcal{N}(18, 0.43)$	14 ± 6	24 ± 19	33 ± 27	41 ± 36	50 ± 33	44 ± 39	52 ± 35
$\mathcal{P}(22.27)$	12 ± 8	26 ± 24	31 ± 25	43 ± 38	49 ± 31	44 ± 38	51 ± 39
$\mathcal{N}(50, 0.69)$	13 ± 7	28 ± 23	33 ± 27	41 ± 35	50 ± 32	41 ± 40	50 ± 38
$\mathcal{P}(7.62)$	19 ± 15	27 ± 22	36 ± 23	46 ± 32	47 ± 30	54 ± 30	56 ± 33
$\mathcal{N}(1000, 0.98)$	11 ± 6	30 ± 23	32 ± 25	41 ± 35	42 ± 26	41 ± 40	53 ± 38
$\mathcal{N}(35, 0.95)$	20 ± 16	30 ± 27	33 ± 21	44 ± 35	54 ± 28	65 ± 26	61 ± 32
$\mathcal{N}(120, 0.94)$	19 ± 15	27 ± 22	36 ± 23	46 ± 32	47 ± 29	54 ± 29	56 ± 33
Référence entraînement	?	?	62	64	69	70	70
Référence test	?	?	20	21	24	30	28

(a) MFCC

Distribution de durée	EBR-24	EBR-12	EBR-6	EBR0	EBR6	EBR12	EBR24
$\mathcal{N}(18, 0.43)$	19 ± 26	39 ± 37	40 ± 28	54 ± 29	72 ± 22	79 ± 23	79 ± 21
$\mathcal{N}(1000, 0.98)$	16 ± 27	40 ± 36	40 ± 28	54 ± 29	71 ± 24	79 ± 23	80 ± 20
$\mathcal{N}(35, 0.95)$	16 ± 26	33 ± 34	49 ± 28	44 ± 31	70 ± 23	78 ± 24	80 ± 20
$\mathcal{N}(120, 0.94)$	16 ± 26	33 ± 35	48 ± 29	44 ± 28	70 ± 23	78 ± 24	79 ± 21
$\mathcal{P}(22.27)$	17 ± 27	40 ± 36	43 ± 27	54 ± 29	71 ± 24	79 ± 23	80 ± 20
$\mathcal{P}(7.62)$	16 ± 26	33 ± 35	48 ± 29	44 ± 28	70 ± 23	78 ± 24	79 ± 21
$\mathcal{P}(11.52)$	16 ± 26	34 ± 36	40 ± 28	44 ± 29	70 ± 25	79 ± 24	78 ± 21
$\mathcal{N}(177, 0.94)$	16 ± 26	34 ± 36	40 ± 28	44 ± 28	70 ± 25	78 ± 24	78 ± 21
$\mathcal{N}(50, 0.69)$	18 ± 26	39 ± 37	40 ± 28	54 ± 29	71 ± 24	79 ± 23	79 ± 21
$\mathcal{N}(4, 0.11)$	16 ± 26	35 ± 36	39 ± 28	45 ± 29	70 ± 25	79 ± 23	79 ± 21
Référence entraînement	?	?	62	64	69	70	70
Référence test	?	?	20	21	24	30	28

(b) Spectrogramme

TABLE B.1 – Résultats de F-mesure par EBR et distribution de durée pour chaque descripteur

Les résultats sont moyennés et affichés en pourcentage, le signe $\pm$ indique la deviation standard. Les cases en rouge indiquent les meilleurs résultats par colonne.

Annexe C

Analyse des transitions HSMM

Pour observer comment l'algorithme incrémental fonctionne, et pour voir si les modèles de durée sont bien considérés, on affiche le résultat d'une segmentation pour différentes distributions de durée, en indiquant les moments où il y a une transition dans le modèle HSMM. La figure C.1 présente les transitions sous forme de traits noirs (on peut se référer à la figure A.2 et la figure A.1 pour observer les densités des distributions de durée considérées). Comme on s'y attend, une espérance très courte va faire transiter le modèle quasiment à chaque instant.

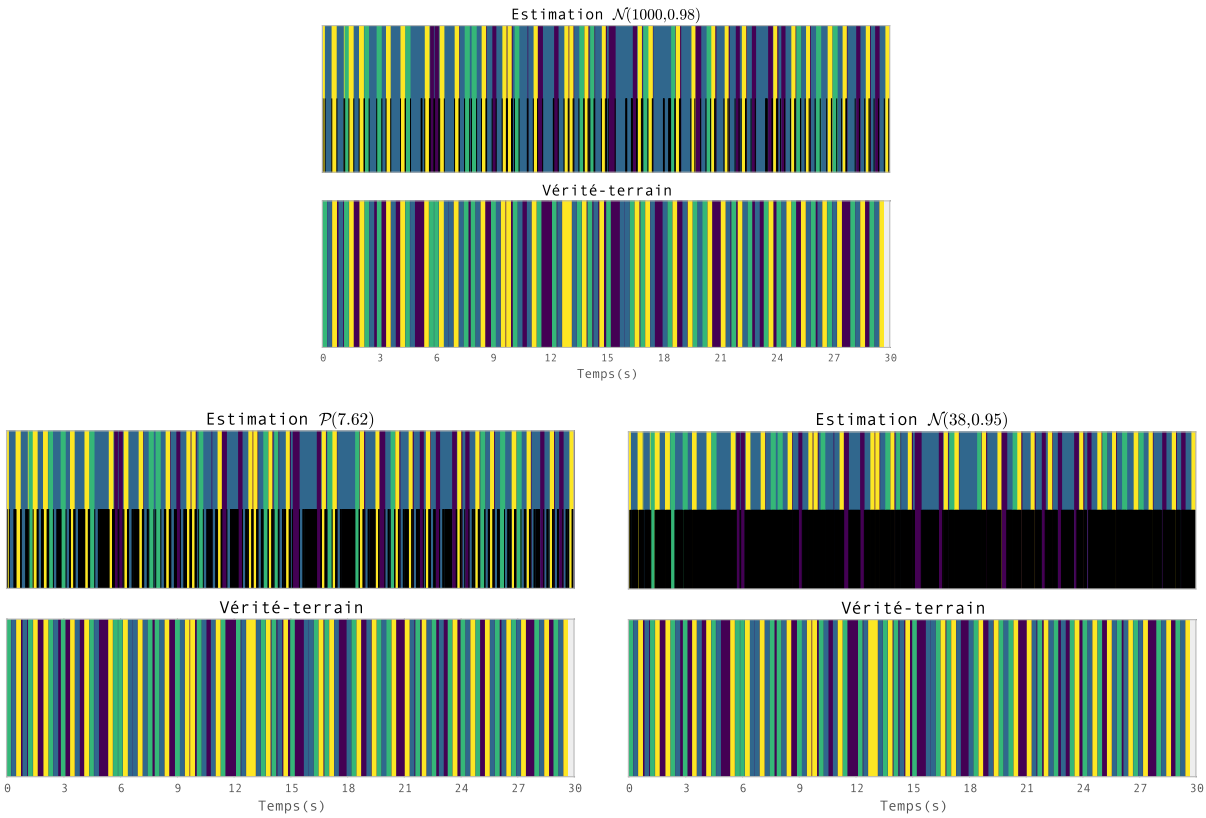


FIGURE C.1 – Observation des transitions d'un HSMM pour trois modèles de durée